



## Article

# Energy-Efficient Virtual Network Embedding: A Deep Reinforcement Learning Approach Based on Graph Convolutional Networks

Peiying Zhang <sup>1,2</sup> , Enqi Wang <sup>1</sup>, Zhihu Luo <sup>1</sup>, Yanxian Bi <sup>3,\*</sup>, Kai Liu <sup>4,5,\*</sup> and Jian Wang <sup>6</sup> 

<sup>1</sup> Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China; zhangpeiying@upc.edu.cn (P.Z.); z23070086@s.upc.edu.cn (E.W.); s21070076@s.upc.edu.cn (Z.L.)

<sup>2</sup> Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250013, China

<sup>3</sup> China Academy of Electronic and Information Technology, CETC Academy of Electronics and Information Technology Group Co., Ltd., Beijing 100041, China

<sup>4</sup> State Key Laboratory of Space Network and Communications, Tsinghua University, Beijing 100084, China

<sup>5</sup> Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China

<sup>6</sup> College of Science, China University of Petroleum (East China), Qingdao 266580, China; wangjiannl@upc.edu.cn

\* Correspondence: biyanxian@cetc.com.cn (Y.B.); liukaiv@tsinghua.edu.cn (K.L.)

**Abstract:** Network virtualization (NV) technology is the cornerstone of modern network architectures, offering significant advantages in resource utilization, flexibility, security, and streamlined management. By enabling the deployment of multiple virtual network requests (VNRs) within a single base network through virtual network embedding (VNE), NV technology can substantially reduce the operational costs and energy consumption. However, the existing algorithms for energy-efficient VNE have limitations, including manual tuning for heuristic routing policies, inefficient feature extraction in traditional intelligent algorithms, and a lack of consideration of periodic traffic fluctuations. To address these limitations, this paper introduces a novel approach that leverages deep reinforcement learning (DRL) to enhance the efficiency of traditional methods. We employ graph convolutional networks (GCNs) for feature extraction, capturing the nuances of network graph structures, and integrate periodic traffic fluctuations as a key constraint in our model. This allows for the predictive embedding of VNRs that is both energy-efficient and responsive to dynamic network conditions. Our research aims to develop an energy-efficient VNE algorithm that dynamically adapts to network traffic patterns, thereby optimizing resource allocation and reducing energy consumption. Extensive simulation experiments demonstrate that our proposed algorithm achieves an average reduction of 22.4% in energy consumption and 41.0% in active substrate nodes, along with a 23.4% improvement in the acceptance rate compared to other algorithms.

**Keywords:** energy-efficient; virtual network embedding; deep reinforcement learning; graph convolutional networks



**Citation:** Zhang, P.; Wang, E.;

Luo, Z.; Bi, Y.; Liu, K.; Wang, J.

Energy-Efficient Virtual Network Embedding: A Deep Reinforcement Learning Approach Based on Graph Convolutional Networks. *Electronics* **2024**, *13*, 1918. <https://doi.org/10.3390/electronics13101918>

Academic Editor: Franco Cicirelli

Received: 2 April 2024

Revised: 2 May 2024

Accepted: 8 May 2024

Published: 14 May 2024



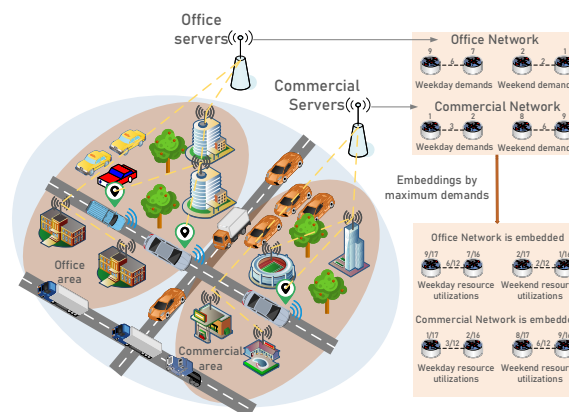
**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

NV technology is one of the key technologies in building the network architectures of the future, designed to improve resource utilization, flexibility, security and management simplicity [1–4]. It enables networks to better adapt to rapidly changing requirements and application scenarios, providing efficient, reliable and personalized network services. In a virtual network environment, multiple applications share a single substrate network, each relatively independent of the other, and each application's requirements are presented as a VNR [5,6]. The process of optimizing the mapping of VNRs to the substrate network

is the main challenge for resource allocation in network virtualization, a challenge that is commonly referred to as the VNE problem.

Most of the existing research on VNE focuses on resource provisioning, providing the maximum hardware support for the life cycle of each VNR [7–11]. However, these resource allocation schemes can result in a large amount of CPU computing power and link bandwidth being underutilized, leading to serious energy wastage and economic loss. For example, a server in a data center runs at 10–50% of its maximum load most of the time [12]. Furthermore, when the server is completely idle, it consumes 70% of its energy under the maximum load [13]. This prompts us to design an energy-efficient and green VNE solution. There are still some problems with the current energy-efficient VNE algorithm. Since the VNE problem is proven to be NP-hard, it is not possible to obtain an optimal solution in a large network setting. To simplify the VNE problem, a large number of scholars have used heuristics to obtain suboptimal solutions in VNE [14]. However, heuristic solutions usually follow a static improvement step manually designed by a dedicated person and still take a long time to run in large-scale networks. Feature extraction is one of the important steps in the VNE process. As the dimensionality and volume of network traffic data continue to increase, the relationships between network nodes and links become increasingly complex. This makes it difficult for traditional methods to obtain network characterization information. For example, the manual extraction of network features is inefficient, and feature extraction by RL directly for networks ignores the spatial characteristics of the network [15]. In order to solve the energy problem in VNE, there are many studies that have sought to find a VNE solution that consumes the smallest amount of energy [16–19]. However, the arrival and departure of VNRs cannot be determined. After a while, their solution may become less energy-efficient. As shown in Figure 1, the network load is heavier in commercial areas on weekends, while the opposite is true in office areas. If network resources are provided at the maximum demand, then a lot of energy is wasted.



**Figure 1.** Network periodic fluctuation model. On weekends, commercial districts see a surge in network loads, while office areas experience a decline.

In this study, we introduce a VNE algorithm underpinned by DRL to address the limitations of existing methodologies. Our proposed approach harnesses deep learning as an automated feature extractor, a technique that has demonstrated remarkable success across various domains, including computer vision [20], natural language processing [21] and voice recognition. Given the NP-hard complexity of VNE, which implies a lack of polynomial-time algorithms capable of solving all instances, especially at scale, traditional methods such as mixed integer programming (MIP) may fall short. This is due to the absence of a known polynomial-time algorithm that can consistently identify optimal solutions across all problem instances. To overcome these challenges, we propose a DRL-based method characterized by its robust adaptability and swift decision-making capabilities. DRL's strength lies in its ability to autonomously learn and extract features, which not only accelerates the computational process through parallelization but also excels in managing

uncertainty and continuous decision spaces. Furthermore, DRL exhibits strong generalization post-training, enabling it to adapt to various network topologies and demand patterns. It adeptly handles large-scale data and high-dimensional features, automatically learning feature representations to extract meaningful information from the data. Reinforcement learning further optimizes the parameters of the deep learning network, enhancing the model's convergence rate.

Within this framework, an agent interacts with the network environment, selecting appropriate embedding actions based on the current state and continually refining its policy to maximize the cumulative rewards. Additionally, we integrate a GCN to automate the extraction of network features, a critical step in VNE problem-solving. Unlike previous methods that depend on manual tuning and heuristic algorithms, a GCN captures the intricate relationships between network nodes and edges through its convolutional layers. This approach minimizes potential biases associated with manual feature extraction. The GCN's suitability for graph-structured data processing allows it to consider the topological interconnections between nodes, offering a distinct advantage over conventional methods. When combined with DRL, the GCN facilitates an end-to-end optimization process that streamlines feature extraction toward embedding decision-making. This integrated approach maintains efficiency as the network sizes increase, making it viable for large-scale VNE problems. The feature representations learned by the GCN generalize well, ensuring that our model performs effectively across diverse network structures and environments.

Moreover, to counteract the degradation of embedding strategies over time, we introduce a mechanism that leverages the periodic fluctuations in network traffic to adjust the embedding strategies dynamically. Timestamps are incorporated into the feature extraction process to analyze and predict traffic fluctuations, providing a foundation for responsive strategy adjustments. The primary contributions of this paper are as follows.

- (1) We use RL as the learning agent and introduce the Asynchronous Advantage Actor-Critic (A3C) algorithm to ensure the efficiency and robustness of sample training and to optimize the learning agent. Experiments show that the learning agent using the A3C algorithm converges faster than with other algorithms.
- (2) We use the GCN-based DL algorithm as the feature extractor for the VNE. In particular, the convolutional kernel of our constructed convolutional network can automatically extract features from the substrate network, providing an important basis for network resource scheduling.
- (3) We add a temporal attribute to the state of the network and simulate the periodic fluctuations of the network. Experiments show that our algorithm can sense the periodic fluctuations in network traffic, providing an important basis for the prediction of the level of the network load.

The rest of the paper is organized as follows. Section 2 describes the VNE algorithm and related work on energy saving. Section 3 presents the VNE model and gives the problem statement. Section 4 describes our energy-efficient VNE algorithm in detail. Section 5 presents and analyzes the simulation experiments and experimental results. Finally, a summary and outlook is given.

## 2. Related Works

In this section, we first introduce VNE-related research, and then we compile recent relevant research in the field of energy-efficient VNE.

### 2.1. Virtual Network Embedding

Early research in VNE focused on designing efficient algorithms to maximize the utilization of substrate resources [22]. The best known example is the node-link two-stage coordination algorithm ViNEYard-VN proposed by Chowdhury et al. [22]. The modified approach describes the VNE problem as a linear program and then designs two online VNE algorithms, D-ViNE and R-ViNE, using deterministic and random rounding techniques, respectively. The results show that these algorithms can effectively improve

the revenue of VNE. In order to reflect the relative importance of different nodes, a Markov Random Wander (RW)-based model has been used, which ranks network nodes according to their resources and topological characteristics [7]. Based on the model, the research introduces two VNE algorithms: the first is a two-stage embedding process that emphasizes the importance of nodes, and the second is a backtracking VNE method that employs a breadth-first search for network traversal. Extensive simulation experiments show that the RW-based algorithm improves the long-term average revenue and acceptance rate of VNE. In response to the huge overhead of embedding in single-domain VNEs, an improved particle swarm algorithm combined with a multi-domain embedding scheme has been introduced into VNE, and experiments show that the scheme can effectively improve resource utilization [23]. With the development of artificial intelligence, many machine learning (ML) algorithms have been used instead of heuristic algorithms to solve VNE problems [24]. To obtain the embedding solution in an acceptable time, the VNE problem has been described as a Markov Decision Process (MDP) and a Monte Carlo Tree Search (MCTS) algorithm has been used to design an action strategy (node mapping) for the proposed MDP [25]. However, MCTS requires a full process to be run for each embedding decision, so this makes the algorithm time-consuming.

## 2.2. Energy-Efficient Virtual Network Embedding

In response to rising energy costs and concerns about the energy consumption of network service infrastructures, Botero et al. first extended the well-known virtual network embedding problem to energy awareness and proposed an MIP [26]. This approach minimizes the overall network consumption of the substrate network by shutting down the remaining unused interfaces and nodes. Due to the NP-hard nature of VNE, the method does not perform well under large-scale networks. Wireless sensor networks (WSNs) have a limited amount of energy available to the sensor nodes. In order to minimize the overall energy consumption, the VNE problem is described as integer linear programming (ILP) [27]. In combination with the so-called E2NE heuristic, a highly efficient VNE is achieved. However, this scheme cannot be applied to large-scale networks due to the limitations of the heuristics. In response to the lack of consideration of network topology information in energy-efficient VNE, field theory-based spectral clustering is applied to extract network features [28]. The modified method builds dynamic regions of interest based on the topological information of the network and then updates the mappable regions in real time, avoiding the local optimum problem. The experiment showed that both the average revenue–cost ratio and the request acceptance rate were improved. However, feature extraction for this method is manual. To alleviate the problem of service degradation due to energy savings, two migration policies were mirrored in VNE [29]. The first is called ‘global’ and is derived by analyzing the context of the day’s traffic and then using a Markov decision process to obtain an embedding solution. The second, referred to as ‘local’, is obtained by observing only the current traffic. The results obtained show that the application of the global policy yields better performance than that of the local policy. Through the analysis, it was found that the state space and state transfer probabilities of the method need to be defined in advance, which may require a lot of computational resources and data for a complex network state space.

## 3. Network Model and Problem Statement

This section meticulously delineates the models of both the virtual network and the substrate network. It then succinctly outlines the problem at hand and articulates the proposed solution, providing a comprehensive overview of the research’s framework.

### 3.1. Substrate Network and Virtual Network Model

The substrate network is formally represented by an undirected graph denoted as  $G^s = \{N^s, L^s, C^s, B^s\}$ , which encapsulates the network’s underlying structure and connectivity.  $N^s = \{n_1^s, n_2^s, \dots, n_{|N^s|}^s\}$  denotes the set of nodes, consisting of the substrate nodes,

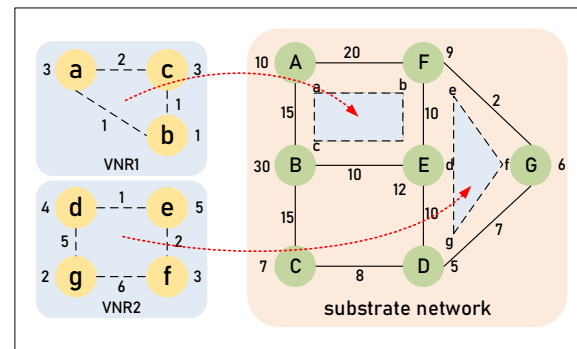
which provide the substrate node resources (e.g., CPU resources, etc.) for virtual network requests. Correspondingly,  $L^s = \{l_1^s, l_2^s, \dots, l_{|L^s|}^s\}$  denotes the set of links, consisting of the substrate links, which provide the substrate link resources (e.g., bandwidth, etc.) for VNRs.  $C^s = \{c_{n_1^s}, c_{n_2^s}, \dots, c_{n_{|N^s|}^s}\}$  denotes the set of CPU resources for the substrate network.  $B^s = \{b_{l_1^s}, b_{l_2^s}, \dots, b_{l_{|L^s|}^s}\}$  denotes the link resources for the substrate network.  $|N^s|$  and  $|L^s|$  are utilized to represent the quantities of substrate nodes and substrate links, respectively.

Similarly, the virtual network can be described as  $G^v = \{N^v, L^v, C^v, B^v\}$ .  $N^v = \{n_1^v, n_2^v, \dots, n_{|N^v|}^v\}$  denotes the set of all virtual nodes, and  $L^v = \{l_1^v, l_2^v, \dots, l_{|L^v|}^v\}$  denotes the set of all virtual links.  $C^v = \{c_1^v, c_2^v, \dots, c_{|N^v|}^v\}$  and  $B^v = \{b_1^v, b_2^v, \dots, b_{|L^v|}^v\}$  denote the set of resource constraints for virtual nodes and virtual links, respectively, where  $c_1^v$  denotes the constraint attribute (e.g., CPU capacity, etc.) of the virtual node with serial number 1, and  $b_1^v$  denotes the virtual link constraint attributes (e.g., link bandwidth, etc.) of the virtual link with serial number 1.  $|N^v|$  and  $|L^v|$  denote the cardinality of the sets  $N^v$  and  $L^v$ , respectively.

### 3.2. Virtual Network Embedding Model

In the realm of VNE, the fusion of the substrate network model and the virtual network model facilitates the representation of a VNR as  $VNR_i = \{G^v, t_a, t_e\}$ , where  $t_a$  signifies the arrival timestamp and  $t_e$  denotes the departure timestamp of the  $i$ th VNR. Upon the reception of a VNR, it is imperative to identify a subset  $G^{s'} \subseteq G^s$  within the substrate network  $G^s$  that can fulfill the VNR's requirements, followed by the deployment of the VNR on the selected subset  $G^{s'}$ . In the event that a suitable subset is not identified, the VNR is consequently rejected. The VNE process is typically bifurcated into two distinct phases: node mapping and link mapping. As delineated in Figure 2, VNR1 and VNR2 are instantiated at disparate locations within the substrate network.

The overarching objective of this endeavor is to enhance the substrate network's capacity to accommodate a continuous influx of VNRs, each accompanied by a unique set of constraints, while simultaneously minimizing the energy consumption and maintaining a robust acceptance rate.



**Figure 2.** Virtual network embedding model. The first virtual network, VNR1, comprises three nodes labeled 'a', 'b', and 'c', interconnected by solid lines with numerical values indicating the link weights. The second virtual network, VNR2, consists of four nodes labeled 'd', 'e', 'f', and 'g', connected in a similar fashion. The underlying substrate network is illustrated in a peach color, with nodes labeled from 'A' to 'G', interconnected by dashed lines, also annotated with link weights. Nodes from the virtual networks are mapped to the substrate network; for instance, node 'a' from VNR1 is mapped to node 'A' of the substrate network, as indicated by red arrows.

### 3.3. Optimization Objective

The energy consumption index emerges as a critical metric for comparison and assessment; within a network, it can be divided into three distinct facets.



- (1) The energy expenditure of the substrate node is directly proportional to its operational load, with the proportionality coefficient denoted as  $p_1$ . The baseline energy consumption is also considered, which is represented by the constant  $p_2$ .
- (2) The constant  $p_2$  is defined as the energy consumption at 50% of the substrate node's peak load, thereby providing a reference point for the node's energy utilization at sub-maximum capacities.
- (3) Additionally, the transition of an actual substrate node from an inactive (off) state to an active (on) state incurs an energy cost, termed the switching cost and quantified by the variable  $E_s$ . Given that the energy consumption at time  $t_s$  is denoted as  $E(t_s)$ , the energy consumption at any other time  $t_e$  can be similarly characterized.

Let  $E(t_s)$  represent the energy consumption at a starting time  $t_s$ ; then, the energy consumption  $E(t_e)$  at current time  $t_e$  can be calculated as

$$E(t_e) = E(t_s) + E_a - E_e \quad (1)$$

where  $E_a$  denotes the increase in energy consumption for VNRs accepted during the time period  $(t_s, t_e)$  and  $E_e$  denotes the decrease in energy consumption for VNRs exited during the time period  $(t_s, t_e)$ . We denote by  $T_s$  the number of substrate nodes required to start the time period  $(t_s, t_e)$  and by  $T_c$  the number of substrate nodes to be closed.  $E_a$  and  $E_e$  are calculated as shown below:

$$E_a = \frac{T_s \cdot E_s}{t_e - t_s} + T_s \cdot p_2 + p_1 \cdot \sum_{i=1}^{|G_a^v|} \sum_{j=1}^{|N_i^v|} c_j^v \quad (2)$$

$$E_e = T_c \cdot p_2 + p_1 \cdot \sum_{i=1}^{|G_e^v|} \sum_{j=1}^{|N_i^v|} c_j^v \quad (3)$$

where  $G_a^v$  and  $G_e^v$  denote the number of VNRs accepted and the number of VNRs exited in time period  $(t_s, t_e)$ , respectively. We express the energy consumption of the current network by calculating the total energy consumption of the substrate nodes, and a smaller  $E(t_e)$  indicates better performance.

Consequently, our primary goal is to identify an optimal VNE strategy, which is designed to efficiently minimize the network's energy consumption, denoted as  $E(t_e)$ , thereby enhancing the overall energy efficiency of the system.

#### 4. Deep Reinforcement Learning Algorithms Based on GCNs in VNE

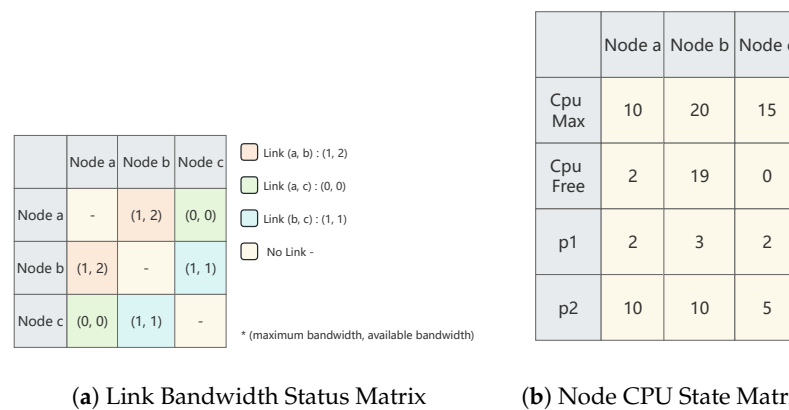
In this section, we first give the definition of reinforcement learning in VNE; then, we describe how to use a GCN to extract feature information from the network; and, finally, we describe the use of the A3C algorithm to enable the training process to converge faster.

##### 4.1. Definition of RL Environment in VNE

RL is a machine learning paradigm that aims to learn how to make optimal decisions in an environment through the interaction between an agent and the environment. In the VNE environment, the agent adopts the corresponding embedding scheme by analyzing the state of the network, and it learns and adjusts the strategy according to the income signal of the embedding scheme. Therefore, RL in the VNE environment consists of four parts: policy, state, action and reward.

##### 4.1.1. Network Status and Policy Definition

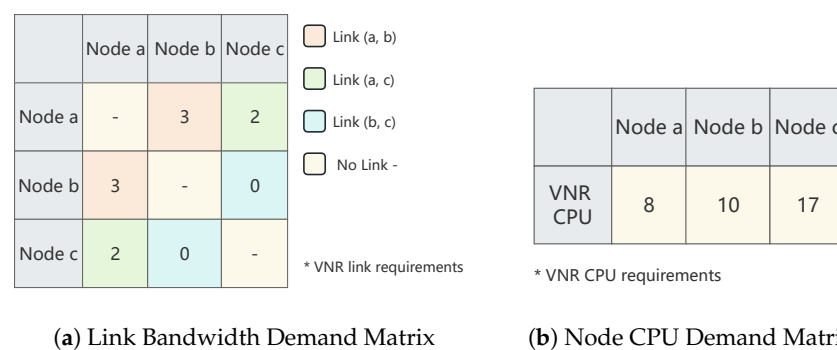
In RL, the state is a description of the environment at a specific moment, which contains information perceived by the agent and is used to make decisions. In a VNE environment, the state is a real-time representation of the network state and serves as input for subsequent feature extraction. The state of the substrate network consists of two matrices, as shown in Figure 3.



**Figure 3.** The state of the substrate network. The figure illustrates the underlying network state comprising three nodes. In this network, the link  $(a, b) : (1, 2)$  has a maximum bandwidth of 2 and an available bandwidth of 1. The term *cpu\_max* denotes the maximum capacity of the underlying node, and *cpu\_free* indicates the available resources of the underlying node. The coefficients *p1* and *p2* represent the energy consumption with an increasing load and the baseline energy consumption, respectively.

The diagram above depicts a substrate network state with three nodes, where link  $(a, b) : (1, 2)$  has a maximum bandwidth of 2 and an available bandwidth of 1. *cpu\_max* represents the maximum capacity of the substrate node, and *cpu\_free* represents the available resources of the substrate node. *p1*, *p2* represent the coefficient of the energy consumption increasing with the load increase and the start-up energy consumption, respectively.

In the context of a VNE, the network state also contains information about the VNR, which is described similarly but is simpler than the substrate network. The VNR does not contain other information, such as the power consumption and load, as shown in Figure 4. The elements of the matrix on the left in Figure 4 represent the VNR link requirements, and the vectors on the right represent the CPU requirements.



**Figure 4.** The demands of the VNR. Within the figure, the matrix on the left delineates the requisite specifications for the VNR link, while the vector positioned to the right corresponds to the CPU requirement.

A policy defines which actions an agent should take in a given state. Specifically, a policy is a function that maps states to the probability distributions of actions.

#### 4.1.2. Network Action Definition

In RL, an action is a decision that an agent can perform in a given state. The choice of action has an important influence on the behavior and strategy of the agent. In the VNE environment, an action is an executable embedding process that assigns the VNR to a subset of the substrate network. Since the number of subsets of the substrate network

risers exponentially with the size of the network, we cannot treat all subsets of the substrate network as actions.

We sort all virtual nodes in breadth-first order and then host them one by one in order to the substrate network, and an action is not complete until the last virtual node is hosted in the substrate network. In the process of the virtual nodes being embedded into the substrate network in an orderly manner, we use the K-shortest path algorithm to solve for the link between two nodes, if it exists.

#### 4.1.3. Network Reward Definition

In RL, the agent's direction of optimization differs from that in traditional supervised learning in that it optimizes the algorithm through feedback obtained from the constant interaction with the external environment. In the VNE environment, in order to maximize the estimated energy consumption, the agent may forgo the currently best-rewarded action in order to obtain better long-term performance.

In this paper, good actions (low energy consumption, meeting resource constraints, etc.) can return positive rewards to increase the probability of the action being selected.

#### 4.2. GCN-Based Deep Learning Algorithms in VNE

In the RL mentioned above, network features are fed into the agent as states, and the agent then outputs a probability distribution of actions based on the states. In order to better implement this process, network features are crucial. Since the network features are constantly changing (with constant VNR additions and departures), we use neural networks with trainable parameters for feature extraction and policy generation, and we use the classical gradient descent method to improve the training parameters.

##### 4.2.1. Feature Extraction in VNE

Within the realm of machine learning, an array of feature extraction methodologies have been introduced, with convolutional neural networks (CNN) and recurrent neural networks (RNN) being prominent examples. CNNs, renowned for their prowess in image processing, employ a series of convolutional and pooling layers to distill local image features, followed by fully connected layers that consolidate these into a comprehensive feature representation. These models excel at capturing the textures, shapes and spatial relationships inherent in image data. RNNs, in contrast, are adept at handling sequential data, adeptly capturing the temporal dynamics characteristic of domains such as stock price forecasting.

However, the application of CNNs and RNNs to graph-structured networks is not straightforward due to the non-Euclidean nature of graph data, which undermines the efficacy of these models designed for Euclidean and temporal data, respectively. To address this, we have adopted the GCN, a paradigm that has emerged as a powerful tool for semi-supervised learning on graph-structured data [30]. GCNs extend the concept of convolution to the graph topology, allowing for the effective capture of the spatial dependencies inherent in the network's structure.

Our use of GCNs confers several advantages over traditional CNN and RNN models. Firstly, GCNs are inherently suited to the non-Euclidean domain of graph data, enabling the more nuanced and accurate extraction of features that reflect the complex interconnections and relationships within the network. This is particularly beneficial for our VNE problem, where the network's topological features are paramount to achieving an efficient and energy-conscious embedding strategy. Furthermore, GCNs automate the feature extraction process, eliminating the need for manual feature engineering, which can be both labor-intensive and biased.

By integrating GCNs into our DRL framework, we have developed a model that not only automates the extraction of topological features but also aligns seamlessly with the reinforcement learning process to optimize network embedding. This unified approach streamlines the optimization process and equips our model with the flexibility to adapt to



varying network conditions and requirements, thereby enhancing the overall performance and scalability of our VNE solution.

We assume that the graph  $G$  has  $m$  nodes and each node has  $n$  features; then, the node features of the graph  $G$  can be represented by a matrix of  $n * m$ . To represent the spatial features of the graph  $G$ , a Laplacian matrix and an orthogonal factorization are used. The Laplace matrix  $L$  is expressed as

$$L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \quad (4)$$

where  $D$  is the degree matrix (diagonal matrix) of the graph  $G$ , and the elements on the diagonal are the degrees of each node in turn.  $A$  is the adjacency matrix of the graph  $G$ .  $I$  is the unit matrix. Considering that  $L$  is semi-positive definite, we factorize  $L$ . Then, the eigenvectors of  $L$  can form an orthogonal basis  $U$  in  $n$  dimensions. We assume that the vector consisting of the features of each node of the graph  $G$  is denoted by  $f$ . The Fourier transformation of the vector  $f$ , represented by  $\hat{f}$ , on graph  $G$  can be computed as

$$f = U \hat{f} \quad (5)$$

According to the convolution theorem, the Fourier transform of the function convolution is the product of the function Fourier transform, and the convolution of  $f$  and the convolution kernel  $h$  on graph  $G$  can be stated as

$$(f * h)G = U \left( (U^T h) \odot (U^T f) \right) \quad (6)$$

where  $\odot$  is the element-wise Hadamard product. By continuously optimizing the parameters in  $h$ , the output will become better and better. In order to extract features that are biased (energy and CPU, etc.), we set up the filter  $h = \sum_{j=0}^K \alpha_j \Lambda^j$ , where  $(\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_{K-1})$  are arbitrary parameters, which is initialized and then adjusted using error direction propagation.  $\Lambda$  is a diagonal matrix with the eigenvalue of  $L$ . Combined with the above, the output of the GCN can be expressed as

$$h_{output} = \sigma \left( \sum_{j=0}^K \alpha_j \Lambda^j f \right) \quad (7)$$

where  $\sigma(\cdot)$  is the activation function.  $K$  indicates the radiation range of a node by other nodes, i.e., the maximum number of hops by which nodes can influence each other. Due to the nature of the network, the characteristics of a node are influenced by all other nodes, and the closer the node is to another one, the stronger the influence. If the  $K$  value is set too large, not only will the algorithm performance be affected but also the nodes located far from the target will have little influence on the target. If the  $K$  value is set too small, nodes close to the target will be ignored. Therefore, we set  $K$  to 3 in comprehensive consideration of this.

#### 4.2.2. Policy Generation

In the previous subsection, we have extracted the feature information of the network using a GCN; we use feature vectors to describe these features and then use softmax to equate these outputs to a probability distribution of the number of substrate network nodes. This narrows down the output without changing the order of the results. The learning agent can select actions based on the probabilities.

To optimize the parameters of the neural network using the embedded feedback data, we chose the Asynchronous Advantage Actor–Critic (A3C) algorithm. It combines the Actor–Critic method and the idea of asynchronous updating to solve the training problem of traditional reinforcement learning algorithms in massively parallel environments. The main idea of the A3C algorithm is that exploration and learning are performed simultaneously in multiple parallel environments, each of which is operated by independent agents, which

share a global neural network model. Each agent learns based on its own experience and updates the learned results to the global model. This asynchronous updating method can increase the diversity of the data, avoid the sample correlation problem in reinforcement learning, and also improve the efficiency of learning. The algorithm consists of an actor network (consisting of a set of parameters  $\theta$ ) and a critic network (consisting of a set of parameters  $\theta_v$ ). Although the actor network and the critic network are very similar, their outputs are quite different. The former is used to generate a set of parameterized strategies  $\pi_{\theta}$ , and the latter is used to generate a set of parameterized estimated values  $v^{\pi_{\theta}}(s_t, \theta_v)$ . The process of the training algorithm is shown in Algorithm 1.

---

**Algorithm 1:** Training process

---

```

1: Initializing global shared network  $\theta$  and  $\theta_v$ ;
2: Initializing global counter  $T = 0$ ;
3: Initializing thread-specific network  $\theta'$  and  $\theta'_v$ ;
4: Initializing thread step count  $t = 1$ ;
5: while  $T < T_{max}$  do
6:   Reset gradients:  $d\theta = 0$  and  $d\theta_v = 0$ ;
7:   Synchronize thread-specific parameters  $\theta' = \theta$  and  $\theta'_v = \theta_v$ ;
8:    $t_{start} = t$ ;
9:   Get initial state  $s_t$ ;
10:  while  $s_t \neq \text{terminal}$  and  $t - t_{start} < t_{max}$  do
11:    Perform  $a_t$  according to policy  $\pi_{\theta'}(a_t|s_t)$ ;
12:    Receive reward  $r_t$  and new state  $s_{t+1}$ ;
13:     $t = t + 1$ ;
14:     $T = T + 1$ ;
15:  end while
16:   $R = V(s_t, \theta'_v)$ ;
17:  for  $i = t - 1$ ;  $i < t_{start}$ ;  $i++$  do
18:     $R = r_i + R$ ;
19:    accumulate gradients  $d\theta$  and  $d\theta_v$ ;
20:  end for
21:  perform asynchronous update of  $\theta$  using  $d\theta$  and of  $\theta_v$  using  $d\theta_v$ ;
22: end while

```

---

#### 4.3. Implementation

In RL, successful embedded programs usually return positive rewards, but this does not mean that all successful programs are considered equally good. In fact, successful programs can be further subdivided into two categories: those that are categorized as “good” and those that are categorized as “generally good”. The need for this distinction stems from our understanding of the complexity of the substrate network and the diversity of VNR objectives.

A “good” solution is one that not only meets the functional constraints of the network (e.g., CPU and bandwidth), but also maintains low energy consumption and a high revenue-to-cost ratio. On the other hand, “generally good” solutions are those that achieve the desired goals but may not be as good as “good” solutions in some aspects (e.g., energy consumption, long-term accumulative revenue). The distinction between “good” and “generally good” solutions is crucial for the optimization of reinforcement learning. This distinction allows us to evaluate and select the best strategies more precisely. The reward is set as follows.

- (1) *Reward Setting for Node:* To improve the acceptance rate of the VNR, we return a positive reward of 100 for successful embedding strategies and a negative reward of −100 for failed embedding strategies. In this case, the algorithm converges more slowly, making it difficult to obtain good embedding schemes within an accept-

able time frame. Therefore, we add a coefficient  $\lambda$  to the reward so that nodes that are embedded later return a higher reward. The specific node reward settings are as follows:

$$R_n = \begin{cases} 100\lambda_i, & \text{if } a_t \text{ is successful,} \\ -100\lambda_i, & \text{otherwise.} \end{cases} \quad (8)$$

where  $\lambda_i = \frac{i}{|N^v|}$  denotes the reward factor of the  $i$ th embedded node.  $|N^v|$  denotes the total number of virtual nodes.  $\lambda_i$  gradually increases from  $\frac{1}{|N^v|}$  to 1. This is because the embedding options of the subsequent nodes may be limited by the locations or paths already chosen by the previous nodes, resulting in a reduction in their available resources. To better accommodate such differences, we guide the RL algorithm to optimize the embedding order by setting different rewards to better target the embedding selection problem of subsequent nodes.

- (2) *Reward Setting for Energy*: The learning agent not only needs to be effectively embedded in the virtual network by selecting successful actions, but also needs to consider the energy consumption level of the actions. In order to realize actions with low energy consumption, we set a corresponding energy consumption reward for each action. By setting higher reward values to encourage learning agents to select actions with lower energy consumption, we motivate the agents to consider the node energy consumption in their embedding decisions and to adopt strategies that can reduce the energy consumption of the network. We therefore add another factor into the reward function:

$$R_e = \begin{cases} \frac{100}{E_s + p2 + p1 \cdot c_i^v}, & \text{if node is off,} \\ \frac{100}{p1 \cdot c_i^v}, & \text{otherwise.} \end{cases} \quad (9)$$

where  $E_s$  denotes the startup energy consumption, and  $p1$  and  $p2$  denote the coefficient of the energy consumption rise with the load and the baseline energy consumption, respectively. In Equation (9), due to the presence of the startup energy consumption, the selection of a substrate node in the off state significantly increases the startup energy consumption  $E_s$  and the baseline energy consumption  $p2$  of the network, and vice versa. We let the reward be inversely proportional to the energy consumption so that actions with low energy consumption are able to obtain higher rewards.

- (3) *Reward Setting for Balance*: The selected action tends to be more likely to be selected again. In this case, the strategy may be fixated on the local optimal solution and unable to discover a better solution. The lack of exploration means that the strategy is unable to explore and try out new actions to adapt to environmental changes, thus limiting its performance from further improvement. To address this problem, we use a Multi-Level Feedback Queue (MLFQ) to record actions, with the rewards of different queues as shown below:

$$R_b = \begin{cases} 100, & \text{if level is 1 queue,} \\ 90, & \text{if level is 2 queue,} \\ 70, & \text{if level is 3 queue.} \end{cases} \quad (10)$$

In accordance with the formulation presented in Equation (10), it is observed that queue 1 is associated with an action reward of 100, queue 2 corresponds to an action reward of 90, and queue 3 is allocated an action reward of 70. To effectively manage the system, we implement a set of three operational rules. Firstly, upon initial entry into the system, actions are systematically assigned to the level 1 queue. Secondly, once an action has been selected a predetermined number of times at a specific level, its priority is adjusted by moving it to a lower-level queue. Lastly, after a designated temporal interval, all actions currently in the queue are reintegrated into the level 1 queue.

The integration of Equation (10) with the aforementioned rules facilitates a strategic reduction in the rewards of highly favored actions. This gradual decrement in rewards enhances the likelihood of selecting actions with lower rewards, thereby circumventing the potential for suboptimal outcomes.

Ultimately, the reward function associated with the action  $a_t$  is delineated as follows:

$$Reward(a_t) = R_n R_e R_b \quad (11)$$

## 5. Performance Evaluation and Analysis

In this section, we first describe the virtual network environment that we used for our evaluation. Then, we compare the proposed algorithm with the current state-of-the-art energy-efficient virtual network mapping algorithms. We aim to demonstrate the superiority of our proposed algorithm for the energy-efficient virtual network mapping problem.

### 5.1. Experimental Setup

Referring to the previous work [31], we use the GT-ITM tool to generate the substrate and virtual network topology in a grid (50 \* 50). The number of substrate nodes is 100 and the node CPU capacity is between 50 and 100. The probability of the existence of links between the substrate nodes is 50% and the link bandwidth is between 50 and 100. Considering the startup energy consumption of the nodes, the default state of the substrate node is off. We categorize VNRs into three sizes: small, medium and large. The number of virtual nodes for these three sizes of VNRs is 2 to 5, 6 to 9 and 10 to 15. The probability of the existence of links between virtual nodes is 50% and the link bandwidth is between 1 and 20. The CPU capacity of the virtual nodes is between 1 and 10. To simulate the periodic fluctuation of the network, the VNR arrives periodically. The life cycle of each VNR in a network fluctuation cycle is exponentially distributed. The main parameters used in the simulation experiment are shown in Table 1. Our simulation experiment is carried out in Anaconda3 + PyCharm3.6.

**Table 1.** Parameter settings.

Parameter	Value
Substrate Nodes	100
Substrate Links	Approximately 2500
Substrate Nodes' CPU	Uniform Distribution [50,100]
Substrate Links' Bandwidth	Uniform Distribution [50,100]
Virtual Nodes	Uniform Distribution [2,15]
Virtual Links	Approximately Uniform Distribution [1,105]
Virtual Nodes' CPU	Uniform Distribution [1,10]
Virtual Links' Bandwidth	Uniform Distribution [1,20]

### 5.2. Comparison Algorithms and Evaluation Indexes

To evaluate our proposed energy-efficient VNE algorithms, we choose the VNE-EA algorithm, based on mixed integer programming [26]; the MCMCF algorithm, which is a dynamic remapping method [32]; and the heuristic-based EA-VNE algorithm [19]. Table 2 lists the key features of these algorithms, including the ideas and advantages of the algorithms.

**Table 2.** Algorithm characterization.

Algorithm	Description
Our algorithm	Utilizes a deep learning approach based on a GCN to extract network features. It employs a reinforcement learning prediction scheme that continually learns from the environment, particularly when the network traffic exhibits periodic fluctuations.
VNE-EA	Formulates the VNE problem as MIP while incorporating energy-related variables.
MCMCF	Balances the load on the substrate network by dynamically reconfiguring the embedding of virtual nodes and links. Linear programming and Markov chains are employed for virtual node selection and remapping, respectively.
EA-VNE	A heuristic algorithm that employs the best match and weighted shortest path approaches during the node mapping and link mapping phases.

In addition, we explore the impact of changing resource requirements on the flexibility of the algorithm. Specifically, we conduct experiments using VNRs of different sizes and evaluate the performance of the algorithm from three perspectives: energy consumption, active nodes, and the acceptance ratio.

### 5.3. Simulation Results and Performance Analysis

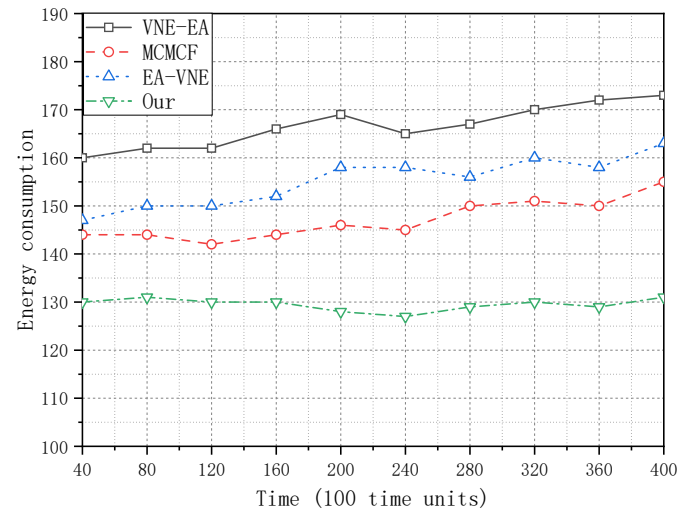
We first compared the instantaneous energy consumption of the algorithms, as shown in Figure 5. It is obvious that our algorithm has lower power consumption than the other three algorithms. Our algorithm allocates the substrate network resources more rationally. VNE-EA performs the worst due to the defects in the mixed integer program. Furthermore, the energy consumption of our algorithm is relatively smooth. On the contrary, the energy consumption of the other three algorithms fluctuates more significantly over time, and all of them reach the highest value at  $Time = 100$ . This indicates that at  $Time = 100$ , a large number of VNR requests arrive, making the energy consumption of the network increase. This is because our algorithm learns the network fluctuations through the interaction of RL with the network and thus computes a more reasonable VNE scheme than the other three algorithms. From the experimental results, at the end of the experiment, our algorithm consumes 24.9%, 21.6% and 20.8% less power than the other algorithms for small, medium and large VNRs, respectively. It is not difficult to find that the relative advantage of our algorithm over others decreases as the VNR size increases. The reason for this is that the resource availability of the substrate network decreases as the VNR size increases, which leads to a decrease in the available optimization space.

We also compared the differences between the four algorithms in terms of active substrate nodes, as shown in Figure 6. The variation curve of the active substrate nodes is similar to the energy consumption, which is due to the fact that the network's energy consumption is positively correlated with the number of active substrate nodes. Overall, our algorithm outperforms the other algorithms. After calculation, we found that the other algorithms had 40.2%, 42.6%, and 40.4% more active substrate nodes than our algorithm for small, medium, and large VNR sizes, respectively. We believe that there are two reasons for this. The first one is due to the fact that the other algorithms use the method of manually extracting features when constructing the network feature model. This method is not only inefficient, but also does not describe the topological features of the network well. On the contrary, our algorithm utilizes a GCN combined with DL to obtain the topological features of the network very well. Second, the stability of the other algorithms is relatively poor.

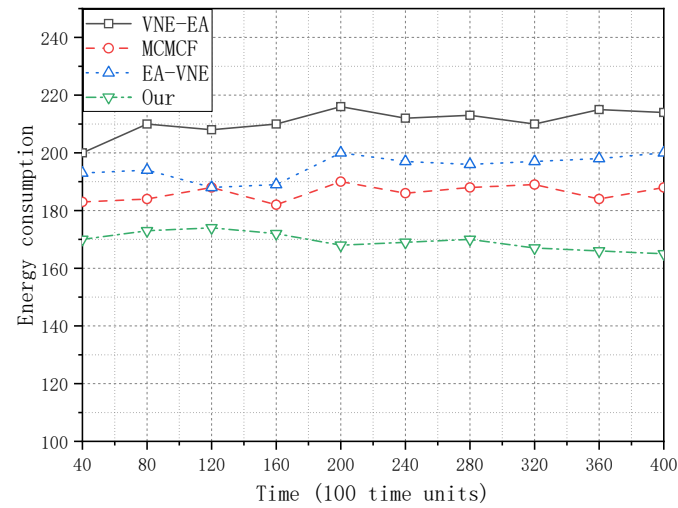
From Figure 6a, we can find that the other three algorithms have a more obvious peak, which indicates that they are more affected by network fluctuations. On the contrary, our algorithm can ensure that the network is relatively stable thanks to the feature of traffic fluctuation awareness. It is worth noting that the fluctuation in the number of active substrate nodes increases as the VNR size increases. This is because the arrival and departure of large-sized VNRs are accompanied by a large number of substrate resource changes.

Smaller-sized VNRs have relatively low resource requirements, resulting in acceptance rates that are not significantly different among the four algorithms. Thus, we only analyze the acceptance rates of the large-sized VNRs, as shown in Figure 7. At the beginning of VNR, all algorithms have relatively high acceptance rates and our algorithm has a slight disadvantage. This is due to the fact that our algorithm considers the maximum return over a future period of time, whereas the other algorithms consider the current maximum return. Nevertheless, our algorithm has a higher acceptance rate in the middle and late stages of VNR, which indicates that our algorithm is more suitable for the allocation of resources. From the experimental results, in the late stage of VNR, our algorithm has a higher acceptance rate than the other algorithms by 40.6%, 10.9% and 18.9%, respectively. This phenomenon is also verified in Figures 5c and 6c. In Figures 5c and 6c, the gap between our algorithm and the other algorithms decreases in the late stage of VNR. This is due to the fact that the acceptance rate of our algorithm is higher than those of the other algorithms, which results in our algorithm occupying more substrate resources.

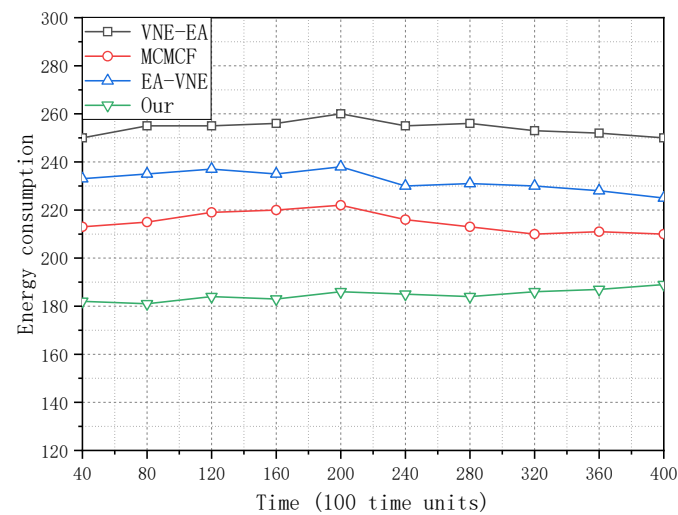




(a) small-sized VNR

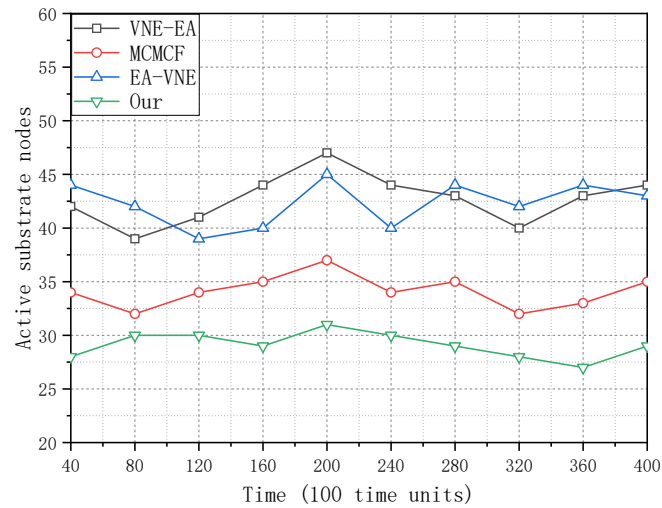


(b) medium-sized VNR

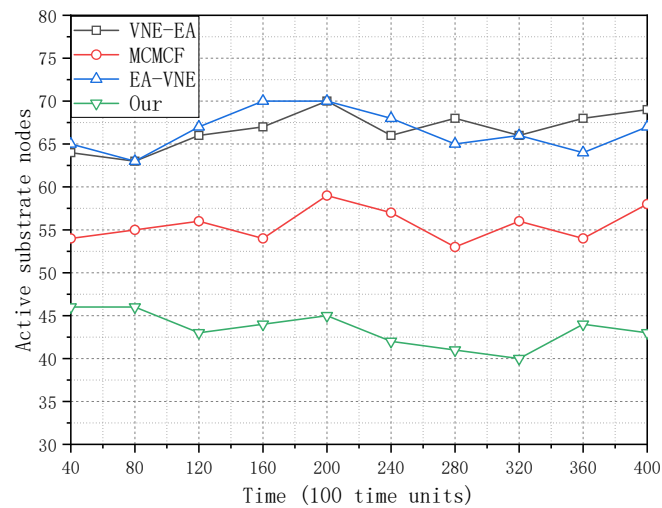


(c) large-sized VNR

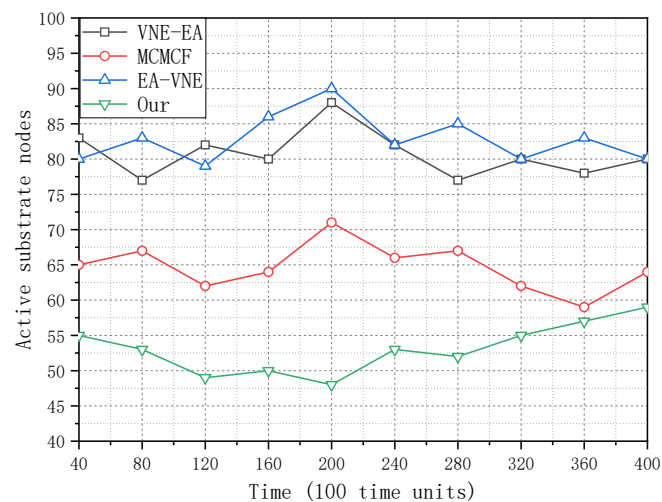
**Figure 5.** Comparison of energy consumption with other algorithms. Our proposed algorithm demonstrates a superior energy efficiency profile when compared to the three alternative algorithms under consideration. This enhanced performance is attributed to its capacity for the more judicious allocation of underlying network resources.



(a) small-sized VNR

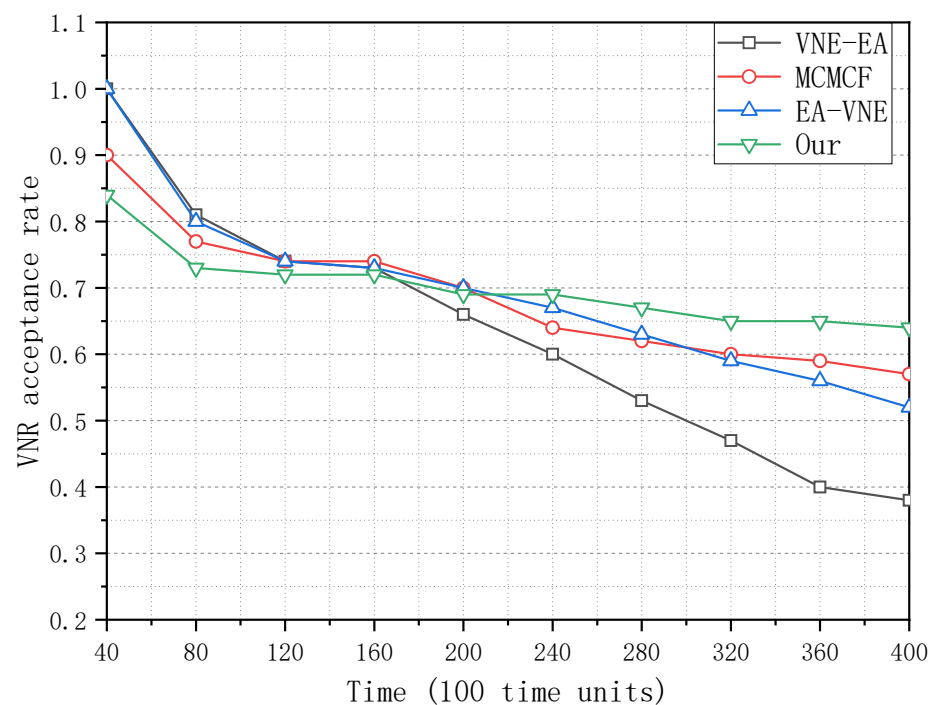


(b) medium-sized VNR



(c) large-sized VNR

**Figure 6.** Comparison of active substrate nodes with other algorithms. We conducted a comparative analysis of the four algorithms with respect to the number of active underlying nodes. The observed trend in the variation in active substrate nodes closely mirrors that of the energy consumption, a phenomenon that can be ascribed to the established positive correlation between the network's energy expenditure and the quantity of active nodes within the substrate.



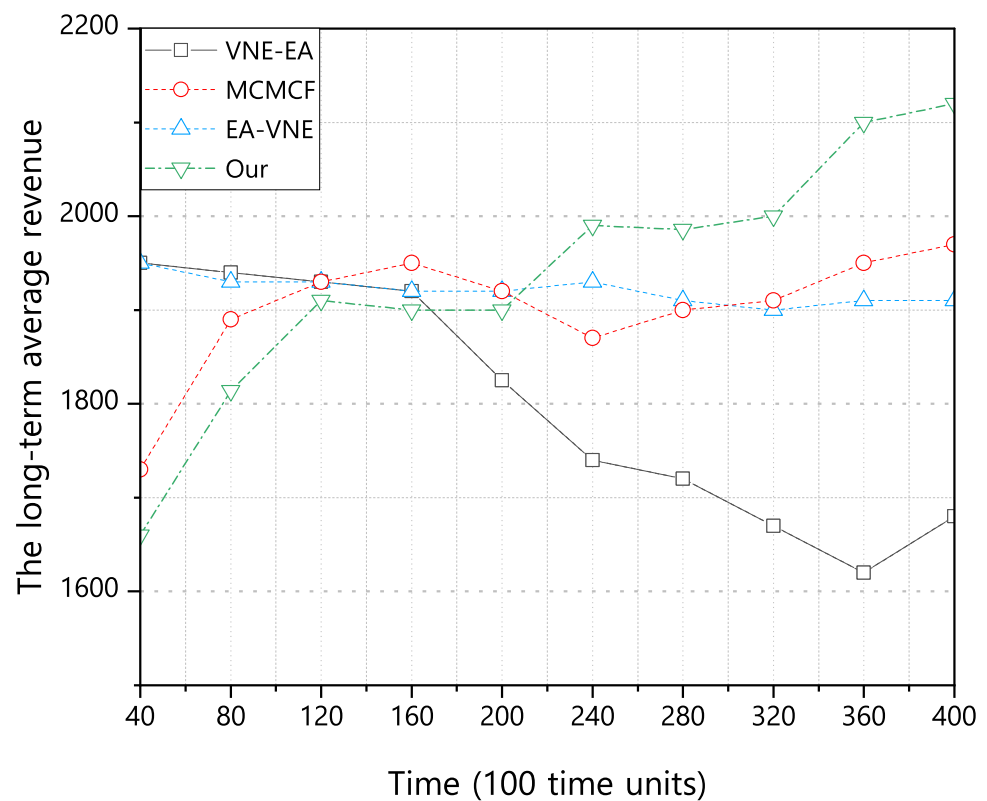
**Figure 7.** Comparison with the VNR acceptance rates of other algorithms. At the onset of the VNR phase, all algorithms show a high acceptance rate, with our proposed algorithm slightly lagging due to its future-oriented payoff optimization. Despite this initial shortfall, our algorithm surpasses others in the mid to late VNR stages, indicating a more rational approach to resource allocation.

Building on our previous discussion regarding the superior performance of our algorithm in reducing the energy consumption and optimizing the resource allocation, we extend our evaluation to include the algorithm's robustness and sustainability in long-term operations. To this end, we introduce the metric of 'long-term average revenue' as a means to assess the comprehensive performance of our algorithm over extended periods of operation.

Figure 8 presents a comparative analysis of the cumulative returns generated by our algorithm alongside existing methods such as VNE-EA, MCMCF, and EA-VNE, over the course of the simulation period. The graph is plotted with time (measured in units of 100 time units) on the x-axis and cumulative payoff on the y-axis, illustrating the sustained performance of each algorithm. Our algorithm, distinguished by the label 'Our', sustains a high level of payoff throughout the simulation, underscoring its stability and reliability in continuous operation.

During the initial phases of the simulation, our algorithm rapidly adapts to the prevailing network conditions, yielding a swift escalation in revenue. This initial performance is followed by a consistently maintained revenue trajectory, indicative of the algorithm's adaptability and robustness amidst the periodic fluctuations in network traffic. Conversely, the VNE-EA algorithm exhibits sluggish revenue growth when scaled to larger networks, a limitation that may stem from its foundational MIP approach. While MCMCF demonstrates strengths in dynamic remapping, it does not outperform our algorithm in terms of long-term cumulative gains. The EA-VNE algorithm, though initially promising, shows a modest decline in the continuity of long-term revenue generation.

These observations suggest that our algorithm offers not only a technically proficient solution but also presents significant benefits in terms of operational stability and sustainability. The graph in Figure 8 corroborates these advantages, further attesting to the algorithm's comprehensive and pragmatic utility in addressing the complexities of the VNE problem.



**Figure 8.** Comparison with the long-term average revenues of other algorithms. These observations indicate that our proposed algorithm delivers a technically adept solution while also conferring substantial benefits in terms of operational stability and sustainability.

#### 5.4. Adaptability Analysis to Data and Environmental Variations

When considering the adaptability of our proposed DRL and GCN-based VNE algorithm to changes in data and environmental conditions, we acknowledge the dynamic and complex nature of real-world network environments, which can significantly impact algorithmic performance. To evaluate the performance of our algorithm under various conditions, we designed a series of simulation experiments to mimic different network loads and data traffic patterns.

In our experiments, particular attention was given to the periodic fluctuations in network traffic and how this information could be leveraged to optimize the embedding strategies. By incorporating a temporal attribute into the network state, our algorithm demonstrated the ability to sense periodic fluctuations in network traffic and adjust the embedding strategies accordingly. Additionally, we considered different sizes of VNRs to simulate a range of environmental conditions and data demands.

Figure 6 indicates that our algorithm exhibits robustness in the face of changing network conditions. Although the performance may be affected under certain extreme scenarios, such as a surge in VNRs arriving simultaneously, overall, the algorithm can quickly adapt to these variations while maintaining low energy consumption and efficient resource allocation.

However, we also recognize that the algorithm may encounter challenges when dealing with unknown or unseen data patterns. To enhance the algorithm's adaptability and generalization capabilities, future work will focus on developing more advanced feature extraction techniques and exploring the use of unsupervised or semi-supervised learning methods to improve the algorithm's adjustment speed to novel environments.

## 6. Conclusions

While notable advancements have been achieved in VNE, the domain continues to face challenges, such as protracted solution times, multi-objective optimization, and dynamic network adaptation. This study introduces a DRL algorithm, underpinned by a GCN, designed to address the VNE challenge. Our proposed algorithm aims to fulfill dual objectives: augmenting the performance and promoting energy efficiency. Through the integration of the GCN, we have significantly enhanced the feature extraction capabilities, thereby bolstering the algorithm's efficiency. Furthermore, this research contributes to network resilience against peak traffic fluctuations by incorporating temporal elements into network feature considerations. We leverage the asynchronous A3C algorithm within an RL framework to facilitate parallel agent training. This methodology encompasses pivotal processes such as policy formulation and model development. The parallel training scheme instituted not only elevates the algorithm's efficacy but also expedites the learning trajectory. Empirical evaluations underscore that our algorithm outperforms comparative models by demonstrating lower energy consumption and a reduced count of active substrate nodes across varying VNR dimensions. Additionally, our algorithm exhibits superior adaptability to peak VNR scenarios, underscoring its robustness and potential for practical applications in VNE.

Despite the promising results, this study has certain limitations. The algorithm's performance has been evaluated in controlled scenarios, and its effectiveness in real-world environments remains to be verified. The reliance on simulated data for training and evaluation may not capture the full complexity of the actual network conditions. Additionally, the computational demands of the algorithm could pose deployment challenges in resource-constrained settings.

In light of these limitations, future work should concentrate on refining the DRL algorithm to reduce the computational complexity and enhance the scalability. Investigating the integration of other machine learning techniques to bolster the algorithm's predictive accuracy is also recommended. Furthermore, developing a more comprehensive benchmarking framework to assess VNE algorithms under a broader spectrum of network conditions would be invaluable.

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

**Funding:** This work is partially supported by the Natural Science Foundation of Shandong Province under Grants ZR2023LZH017 and ZR2022LZH015, is partially supported by the Open Foundation of the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Qilu University of Technology (Shandong Academy of Sciences) under Grant 2023ZD010 and is partially supported by the National Natural Science Foundation of China under Grant 62341130.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Conflicts of Interest:** Author Yanxian Bi was employed by the company CETC Academy of Electronics and Information Technology Group Co. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## References

1. Wang, A.; Iyer, M.; Dutta, R.; Rouskas, G.N.; Baldine, I. Network virtualization: Technologies, perspectives, and frontiers. *J. Light. Technol.* **2012**, *31*, 523–537. [[CrossRef](#)]
2. Liang, C.; Yu, F.R. Wireless Network Virtualization: A Survey, Some Research Issues and Challenges. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 358–380. [[CrossRef](#)]



3. Chowdhury, N.M.M.K.; Boutaba, R. Network virtualization: State of the art and research challenges. *IEEE Commun. Mag.* **2009**, *47*, 20–26. [\[CrossRef\]](#)
4. Chowdhury, N.M.M.K.; Boutaba, R. A survey of network virtualization. *Comput. Netw.* **2010**, *54*, 862–876. [\[CrossRef\]](#)
5. Khan, I.; Jafrin, R.; Errounda, F.Z.; Glitho, R.H.; Crespi, N.; Morrow, M.; Polakos, P. A Data Annotation Architecture for Semantic Applications in Virtualized Wireless Sensor Networks. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015.
6. Nkomo, M.; Hancke, G.P.; Abu-Mahfouz, A.M.; Sinha, S.; Onumanyi, A.J. Overlay Virtualized Wireless Sensor Networks for Application in Industrial Internet of Things: A Review. *Sensors* **2018**, *18*, 3215. [\[CrossRef\]](#) [\[PubMed\]](#)
7. Cheng, X.; Su, S.; Zhang, Z.; Wang, H.; Yang, F.; Luo, Y.; Wang, J. Virtual network embedding through topology-aware node ranking. *Comput. Commun. Rev.* **2011**, *41*, 38–47. [\[CrossRef\]](#)
8. Chowdhury, N.M.M.K.; Rahman, M.R.; Boutaba, R. Virtual Network Embedding with Coordinated Node and Link Mapping. In Proceedings of the INFOCOM 2009, 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 783–791. [\[CrossRef\]](#)
9. Yu, M.; Yi, Y.; Rexford, J.; Chiang, M. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 17–29. [\[CrossRef\]](#)
10. Zhu, Y.; Ammar, M.H. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. In Proceedings of the INFOCOM 2006, 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, Barcelona, Spain, 23–29 April 2006. [\[CrossRef\]](#)
11. Zhang, S.; Wu, J.; Lu, S. Virtual network embedding with substrate support for parallelization. In Proceedings of the 2012 IEEE Global Communications Conference, GLOBECOM 2012, Anaheim, CA, USA, 3–7 December 2012; pp. 2615–2620. [\[CrossRef\]](#)
12. Barroso, L.A.; Hölzle, U. The Case for Energy-Proportional Computing. *Computer* **2007**, *40*, 33–37. [\[CrossRef\]](#)
13. Fan, X.; Weber, W.; Barroso, L.A. Power provisioning for a warehouse-sized computer. In Proceedings of the 34th International Symposium on Computer Architecture (ISCA 2007), San Diego, CA, USA, 9–13 June 2007; pp. 13–23. [\[CrossRef\]](#)
14. Fischer, A.; Botero, J.F.; Beck, M.T.; de Meer, H.; Hesselbach, X. Virtual Network Embedding: A Survey. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1888–1906. [\[CrossRef\]](#)
15. Yan, Z.; Ge, J.; Wu, Y.; Li, L.; Li, T. Automatic Virtual Network Embedding: A Deep Reinforcement Learning Approach With Graph Convolutional Networks. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1040–1057. [\[CrossRef\]](#)
16. Amokrane, A.; Zhani, M.F.; Langar, R.; Boutaba, R.; Pujolle, G. Greenhead: Virtual Data Center Embedding across Distributed Infrastructures. *IEEE Trans. Cloud Comput.* **2013**, *1*, 36–49. [\[CrossRef\]](#)
17. Zhani, M.F.; Zhang, Q.; Simon, G.; Boutaba, R. VDC Planner: Dynamic migration-aware Virtual Data Center embedding for clouds. In Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium, 27–31 May 2013; pp. 18–25.
18. Su, S.; Zhang, Z.; Liu, A.X.; Cheng, X.; Wang, Y.; Zhao, X. Energy-Aware Virtual Network Embedding. *IEEE/ACM Trans. Netw.* **2014**, *22*, 1607–1620. [\[CrossRef\]](#)
19. Su, S.; Zhang, Z.; Cheng, X.; Wang, Y.; Luo, Y.; Wang, J. Energy-aware virtual network embedding through consolidation. In Proceedings of the 2012 Proceedings IEEE INFOCOM Workshops, Orlando, FL, USA, 25–30 March 2012; pp. 127–132. [\[CrossRef\]](#)
20. Girshick, R.B. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, 7–13 December 2015; pp. 1440–1448. [\[CrossRef\]](#)
21. Kim, Y. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, 25–29 October 2014; pp. 1746–1751. [\[CrossRef\]](#)
22. Chowdhury, M.; Rahman, M.R.; Boutaba, R. ViNEyard: Virtual Network Embedding Algorithms with Coordinated Node and Link Mapping. *IEEE/ACM Trans. Netw.* **2012**, *20*, 206–219. [\[CrossRef\]](#)
23. 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. [\[CrossRef\]](#)
24. Yao, H.; Zhang, B.; Zhang, P.; Wu, S.; Jiang, C.; Guo, S. RDAM: A Reinforcement Learning Based Dynamic Attribute Matrix Representation for Virtual Network Embedding. *IEEE Trans. Emerg. Top. Comput.* **2021**, *9*, 901–914. [\[CrossRef\]](#)
25. Haeri, S.; Trajkovic, L. Virtual Network Embedding via Monte Carlo Tree Search. *IEEE Trans. Cybern.* **2018**, *48*, 510–521. [\[CrossRef\]](#) [\[PubMed\]](#)
26. Botero, J.F.; Hesselbach, X.; Duelli, M.; Schlosser, D.; Fischer, A.; de Meer, H. Energy Efficient Virtual Network Embedding. *IEEE Commun. Lett.* **2012**, *16*, 756–759. [\[CrossRef\]](#)
27. Raei, V.M.; Ebrahimzadeh, A.; Rayani, M.; Glitho, R.H.; Barachi, M.E.; Belqasmi, F. Energy Efficient Virtual Network Embedding in Virtualized Wireless Sensor Networks. In Proceedings of the 19th IEEE Annual Consumer Communications & Networking Conference, CCNC 2022, Las Vegas, NV, USA, 8–11 January 2022; pp. 187–192. [\[CrossRef\]](#)
28. He, M.; Zhuang, L.; Tian, S.; Wang, G.; Zhang, K. DROI: Energy-efficient virtual network embedding algorithm based on dynamic regions of interest. *Comput. Netw.* **2020**, *166*, 106952. [\[CrossRef\]](#)
29. Eramo, V.; Miucci, E.; Ammar, M.H. Study of Reconfiguration Cost and Energy Aware VNE Policies in Cycle-Stationary Traffic Scenarios. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 1281–1297. [\[CrossRef\]](#)
30. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.

31. Zhang, P.; Pang, X.; Bi, Y.; Yao, H.; Pan, H.; Kumar, N. DSCD: Delay Sensitive Cross-Domain Virtual Network Embedding Algorithm. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 2913–2925. [[CrossRef](#)]
32. Gao, L.; Rouskas, G.N. Virtual Network Reconfiguration with Load Balancing and Migration Cost Considerations. In Proceedings of the 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 16–19 April 2018; pp. 2303–2311. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.