Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8889923, 13 pages
https://doi.org/10.1155/2021/8889923

Research Article

Energy-Efficient Virtual Network Embedding Algorithm Based on Hopfield Neural Network

Mengyang He,[1] **Lei Zhuang**,[1] **Sijin Yang**,[1] **Jianhui Zhang**,[2] **and Huiping Meng**[3]

[1]*School of Information and Engineering, Zhengzhou University, Zhengzhou, Henan, China*
[2]*China National Digital Switching System Engineering & Technological R&D Center, China*
[3]*State Grid Henan Information & Telecommunication Company (Data Center), China*

Correspondence should be addressed to Lei Zhuang; ielzhuang@zzu.edu.cn

Received 30 April 2020; Revised 29 October 2020; Accepted 13 January 2021; Published 28 January 2021

Academic Editor: Javier Prieto

To solve the energy-efficient virtual network embedding problem, this study proposes an embedding algorithm based on Hopfield neural network. An energy-efficient virtual network embedding model was established. Wavelet diffusion was performed to take the structural feature value into consideration and provide a candidate set for virtual network embedding. In addition, the Hopfield network was used in the candidate set to solve the virtual network energy-efficient embedding problem. The augmented Lagrangian multiplier method was used to transform the energy-efficient virtual network embedding constraint problem into an unconstrained problem. The resulting unconstrained problem was used as the energy function of the Hopfield network, and the network weight was iteratively trained. The energy-efficient virtual network embedding scheme was obtained when the energy function was balanced. To prove the effectiveness of the proposed algorithm, we designed two experimental environments, namely, a medium-sized scenario and a small-sized scenario. Simulation results show that the proposed algorithm achieved a superior performance and effectively decreased the energy consumption relative to the other methods in both scenarios. Furthermore, the proposed algorithm reduced the number of open nodes and open links leading to a reduction in the overall power consumption of the virtual network embedding process, while ensuring the average acceptance ratio and the average ratio of the revenue and cost.

1. Introduction

With the emergence of new network applications and the diversification of business requirements, the gap between the limited physical resources of the existing network architectures and the increasing user demands has become increasingly prominent. Network virtualization has boosted the application and development of new technologies and concepts including software-defined networks [1], network function virtualization [2], and the Internet of Things [3], to some extent, thus, alleviating the aforementioned problem in the existing network structure [4].

Network virtualization technology allows heterogeneous virtual networks to coexist on the same substrate network and divides the Internet service providers in the traditional network service model into two independent roles: the substrate network resources are managed and maintained by the infrastructure providers. The creation and management of the virtual networks are handled by the service providers [5]. The need for reasonable and effective services for users has increased the importance of the design of virtual network embedding (VNE) algorithms. Each virtual network can be regarded as a resource slice containing virtual nodes and virtual links of the substrate network. The problem of allocating the substrate network resources to a virtual network with node and link resource constraints is called the VNE problem. With the development of new concepts, such as green networks, allocating substrate resources in an energy-efficient way is an urgent issue that needs to be solved.

The VNE problem is a discrete combinatorial optimization problem and has been proven to be a non-deterministic polynomial-time hardness (NP-hard) problem [6]. Common solutions to this problem mainly involve heuristic algorithms and exact algorithms [7].

Several studies have employed heuristic algorithms. Liu et al. [8] established a mixed-integer linear programming model for the VNE problem and proposed a VNE algorithm on the basis of node attribute evaluation. In the node embedding stage, the node resource richness, security, and topology connection are taken into account. In the link mapping phase, the bandwidth and path hops are comprehensively considered to obtain the mapped path. This method fixes the attributes during the embedding process and ignores the other node and link attributes that affect the embedding performance. Liu et al. [9] proposed a cluster-based collaborative embedding algorithm to optimize the energy consumption. A weighted graph is generated according to the virtual network topology, and it is then clustered with a local clustering algorithm. An embedding scheme is finally obtained on the basis of the clustering. This scheme depends on the cluster size, which is likely to exacerbate the fragmentation of the substrate networks. He et al. [10] proposed a heuristic VNE algorithm on the basis of the reinforcement learning and curiosity mechanism; the algorithm regards the energy efficiency and request acceptance rate as the main optimization goals. In addition, it considers other potential optimization goals in depth while improving the energy savings and the request acceptance rates. This solution has a high calculation complexity, and it is not suitable for large-scale network environments. Zhuang et al. [11] proposed a VNE algorithm on the basis of the diffusion wavelets. With the diffusion wavelets, the algorithm comprehensively evaluates the topological structure of the nodes, the connection density, and the number of links between the nodes. Several studies have employed evolutionary computation algorithm like the ant colony system (ACS) algorithm. Wang et al. [12] proposed an ant colony system-based VNE algorithm, for the online VNE problem. Different from previous work that only considers the resource of nodes in node embedding phase, they take the distance message related to links into consideration so that they can reduce the cost of VN requests. Zheng et al. [13] proposed a link resource heuristic information and incorporate it into the search process of ACS. They sort the embedding sequence of virtual nodes according to their link resources. ACS is used to embed the virtual nodes onto substrate nodes according the sorted sequence, while the virtual links are mapped via a shortest path strategy for the embedded nodes. Chen et al. [14] model the cloud workflow scheduling as a multiobjective optimization problem that optimizes both execution time and execution cost. A multiobjective ant colony system based on a coevolutionary multiple populations for multiple-objective framework is proposed, which adopts two colonies to deal with these two objectives, respectively. Liu et al. [15] applied the evolutionary computing in virtual machine placement to minimize the number of active physical servers, so as to schedule underutilized servers to save energy. It effectively minimizes the number of active servers used for the assignment of virtual machines (VMs) from a global optimization perspective through a strategy for pheromone deposition which guides the artificial ants toward promising solutions that group candidate VMs together.

Although the above schemes solve the VNE problem to a certain extent, most of them rely on artificial rules, and they usually sort the nodes according to the remaining resources and relative distances. The parameters are fixed, and they cannot be optimized, and the embedding process tends to fall into a local optimum.

Other studies have employed exact algorithms. Chemodanov et al. [16] proposed a constrained shortest path algorithm that is accurate for the VNE problem. This algorithm uses search space reduction technology, and it theoretically achieves a secondary acceleration at an order of magnitude that is faster than the branch and bound solution. This solution lacks consideration of the node and link attributes, thus, causing excessive waste of the substrate resources in the actual embedding process. Mijumbi et al. [17] proposed a VNE scheme on the basis of the column generation algorithm. This scheme formalizes the VNE problem into path-based mathematical programming, and it is proposed as a dual problem. By solving the dual problem of the VNE, a solution to the original problem is generated. Wang et al. [18] proposed a path-based compact integer linear programming model. Based on the branch pricing method, a column generation process was introduced to solve the proposed integer linear programming model. Then, the approximate optimal solution of the VNE problem was obtained. Yang et al. [19] addressed the problem in which the constraint conditions include only the node capacity and link bandwidth in the VNE. An improved exact VNE algorithm was proposed, and an integer linear programming was established to embed the virtual networks with the position constraints. This scheme limits the location of the links, thus lacking universality. Houidi et al. [20] comprehensively considered the energy savings, the high availability, and load balancing. An accurate adaptive embedding algorithm was proposed to deal with the node and link failures in the cloud environment. An embedding scheme was obtained by establishing a mixed-integer programming model for the VNE.

Exact algorithms usually convert the VNE problem into a path-based integer programming problem or a mixed-integer programming problem. Hence, the calculation complexity of the solution is too high for it to be suitable for solving large-scale problems or those with a high link density. As the work scale of the problem increases, the effectiveness of the optimization algorithm significantly decreases.

There are other studies that have focused on the migration algorithm. Eramo et al. [21] proposed a methodology based on Markov decision processes to evaluate the optimal migration policies that consider the operation (in terms of the energy consumption) and the reconfiguration costs. Two migration policies were introduced and compared in Eramo et al. [22]. The first migration policy, which is referred to as "global," is based on the knowledge of the entire daily traffic profile and is determined with a Markov decision process. The second, referred to as "local," is only based on the knowledge of the current traffic. This study works on the embedding problem with time-varying resource requirements of the virtual network. Our research assumes that the arrival of the virtual network obeys the Poisson distribution, and the node and link resources obey the uniform distribution, which is fixed during the existence period. This implies

that the virtual network traffic does not change with time. The research background for this type of algorithm is different from the one considered in this study; hence, it is not discussed herein.

In view of the above problems, this study introduces the Hopfield network to solve the VNE problem. This study is aimed at reducing the energy consumption in the VNE process. We first considered the structural feature value to select a candidate set for virtual network embedding. Next, we used the Hopfield network to solve the energy-efficient embedding problem in the candidate set, and we automatically updated the network parameters through training and updating the neural network. As a result of the characteristics of the parallel computations for the Hopfield neural network, the number of calculations did not increase exponentially when the number of dimensions increased. This work considers optimizing the energy consumption in the VNE and proposes an energy-efficient VNE algorithm on the basis of the Hopfield neural network (VNEE_HNN). This solution takes the energy consumption in the VNE as the optimization goal and builds a model for it. To solve the original problem, we applied the augmented Lagrange method (ALM) to transform the VNE constraint problem into an unconstrained problem. Furthermore, an iterative data flow that consists of virtual network node embedding binary variables and Lagrangian multipliers was constructed. The unconstrained problem was transformed into an energy function and was used as input for the Hopfield network. The variables in the virtual network node embedding problem were mapped to the neuron state of the neural network. The network weights were automatically updated by minimizing the energy function. The iterations were updated until the energy function of the neural network converged to a minimum value. In other words, when the neural network approached equilibrium, a virtual network node embedding scheme was generated. After the node positions were determined, the shortest path method was used to determine the link positions, and an energy-efficient VNE scheme was obtained.

In summary, the main contributions for this study are as follows: (1) apply the wavelet diffusion to take the structural feature value into consideration; (2) convert the virtual network energy conservation constraint problem to an unconstrained problem, and (3) use the Hopfield network to solve the virtual network energy conservation embedding unconstrained problem.

2. Energy-Efficient VNE Modeling

The total energy consumption is composed of nodes and links. The calculation of the node energy consumption is shown in

$$\mathrm{EN}_i = \left(p_{\mathrm{b}} + p_{\mathrm{l}} \cdot u^{n_i^{\mathrm{s}}}\right) f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}}, \tag{1}$$

where p_{b} is the basic energy consumption of the node, p_{l} represents the scale factor of the energy consumption, p_{m} is the maximum energy consumption of the node ($p_{\mathrm{l}} = p_{\mathrm{m}} - p_{\mathrm{b}}$), and $u^{n_i^{\mathrm{s}}}$ indicates the CPU utilization of the node n_i^{s} ($u^{n_i^{\mathrm{s}}} = \mathrm{ReqCPU}(n_i^{\mathrm{v}})/\mathrm{CPU}(n_i^{\mathrm{s}})$). $\mathrm{ReqCPU}(n_i^{\mathrm{v}})$ is the request resource of the virtual node n_i^{v}, and $\mathrm{CPU}(n_i^{\mathrm{s}})$ is the total number of resources for the substrate node n_i^{s}. $f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}}$ is a binary variable for the node. In other words, when a virtual node is mapped onto a substrate node, the value is 1; otherwise, the value is 0.

The link energy consumption can be calculated by applying

$$\mathrm{EL}_i = f_{l_j^{\mathrm{s}}}^{l_j^{\mathrm{v}}} \cdot p_n, \tag{2}$$

where p_n is a constant and denotes the basic energy consumption of the link. $f_{l_j^{\mathrm{s}}}^{l_j^{\mathrm{v}}}$ is a link binary variable; that is, when the virtual link is mapped onto the substrate links, the value is 1; otherwise, it is 0.

Based on the energy consumption of the nodes and links, an energy consumption model of VNE is established, as demonstrated below.

Objective function:

$$\min \sum_{\forall n_i^{\mathrm{v}} \in N^{\mathrm{V}}, n_i^{\mathrm{s}} \in N^{\mathrm{S}}} \mathrm{EN}_i + \sum_{\forall n_i^{\mathrm{v}}, n_j^{\mathrm{v}} \in N^{\mathrm{V}}, n_u^{\mathrm{s}}, n_w^{\mathrm{s}} \in N^{\mathrm{S}}} \mathrm{EL}_i, \tag{3}$$

Capacity constraints:

$$\left(\forall n_i^{\mathrm{v}} \in N^{\mathrm{V}}\right)\left(\forall n_i^{\mathrm{s}} \in N^{\mathrm{S}}\right): f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}} \cdot \mathrm{ReqCPU}(n_i^{\mathrm{s}}) \le \mathrm{ReCPU}(n_i^{\mathrm{s}}), \tag{4}$$

$$\left(\forall l_j^{\mathrm{v}} \in L^{\mathrm{V}}\right)\left(\forall l_j^{\mathrm{s}} \in L^{\mathrm{S}}\right): f_{l_j^{\mathrm{s}}}^{l_j^{\mathrm{v}}} \cdot \mathrm{ReqBW}\left(l_j^{\mathrm{v}}\right) \le \mathrm{ReBW}\left(l_j^{\mathrm{s}}\right), \tag{5}$$

Variable constraints:

$$\left(\forall n_i^{\mathrm{v}} \in N^{\mathrm{V}}\right): \sum_{\left(\forall n_i^{\mathrm{v}} \in N^{\mathrm{S}}\right)} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}} = 1, \tag{6}$$

$$\left(\forall n_i^{\mathrm{s}} \in N^{\mathrm{S}}\right): \sum_{\left(\forall n_i^{\mathrm{v}} \in N^{\mathrm{S}}\right)} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}} \le 1, \tag{7}$$

$$\left(\forall n_i^{\mathrm{v}} \in N^{\mathrm{V}}\right)\left(\forall n_i^{\mathrm{s}} \in N^{\mathrm{S}}\right): f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}} \in \{0, 1\}, \tag{8}$$

$$\left(\forall l_j^{\mathrm{v}} \in L^{\mathrm{V}}\right)\left(\forall l_j^{\mathrm{s}} \in L^{\mathrm{S}}\right): f_{l_j^{\mathrm{s}}}^{l_j^{\mathrm{v}}} \in \{0, 1\}, \tag{9}$$

where N^{V} is the set of the virtual nodes, N^{S} is the set of the substrate nodes, and $\mathrm{ReqCPU}(n_i^{\mathrm{s}})$ is the remaining CPU resources of the substrate node n_i^{s} in Equation (4). L^{V} is the set of virtual links, L^{S} is the set of substrate links, $\mathrm{ReqBW}(l_j^{\mathrm{v}})$ is the request resource of the virtual link l_j^{v} in Equation (5), and $\mathrm{ReBW}(l_j^{\mathrm{s}})$ is the remaining bandwidth resource of the substrate link l_j^{s}. Equation (6) ensures that the same virtual node is mapped to only one substrate node. Equation (7) ensures that all of the virtual nodes for the same virtual network cannot be mapped to the same substrate node.

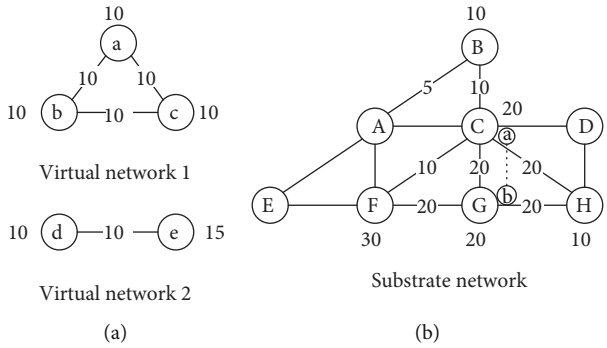

Figure 1: Feature analysis of nodes.

3. Energy-Efficient Algorithm Based on the Hopfield Neural Network

This section proposes an energy-efficient VNE algorithm based on the Hopfield neural network. The algorithm is mainly divided into two parts, including the problem conversion based on the ALM and problem-solving based on the Hopfield neural network. In the process of the problem transformation, we used the augmented Lagrangian multiplier method to transform the VNE constraint problem into an unconstrained problem to facilitate the subsequent solving process. In the problem-solving process, we constructed a dynamic function, used the parallel computing characteristics of the Hopfield neural network, trained the neural network to reach the equilibrium state, and quickly solved the virtual network node embedding problem.

3.1. Initialization. The embedding process of the virtual network needs to consider many factors. Most of the existing studies focus on using the computing power of the substrate nodes or the communication capabilities of the links to improve the embedding performance.

Many factors affect the embedding performance. This section initializes the substrate network according to the nodes' structural characteristics. First, we analyzed the effect of the nodes' structural characteristics on the embedding performance and then used the wavelet diffusion method to take the structural characteristics into consideration. We evaluated the importance of the nodes by calculating the structural feature values (SFV). This divided the candidate set, which contains the area of the most important nodes in the substrate network and uses the Hopfield network to perform energy-saving embedding in this area.

3.1.1. Node Structure Analysis. This paper analyzes the influence of the structural characteristics on the embedding performance in several situations as shown in Figure 1.

As shown in Figure 1(a), the graphs represent two virtual networks. The circles represent the virtual nodes, which are expressed as n_i^{v}, and the numbers next to the circles indicate the amount of request resources of the virtual nodes. The links between the nodes are virtual links, expressed as l_{ij}^{v}. The number on the link represents the request bandwidth. Figure 1(b) shows the substrate network. The substrate nodes are represented as $n_{\mathrm{I}}^{\mathrm{s}}$, the number next to it represents the amount of CPU resources of the substrate node, the substrate link is represented as $l_{\mathrm{IJ}}^{\mathrm{s}}$, and the number on the link represents the bandwidth of the substrate links. Assuming that when virtual network 1 is embedded, a virtual node $n_{\mathrm{a}}^{\mathrm{v}}$ is embedded to the substrate node $n_{\mathrm{C}}^{\mathrm{s}}$, a virtual link $l_{\mathrm{ab}}^{\mathrm{v}}$ is embedded to the substrate link $l_{\mathrm{CG}}^{\mathrm{s}}$, and there are several situations for the virtual node $n_{\mathrm{c}}^{\mathrm{v}}$ embedding process.

Case 1. The number of host nodes affects the embedding performance.

As shown in Figure 1, when the virtual node $n_{\mathrm{c}}^{\mathrm{v}}$ is embedded to the substrate node $n_{\mathrm{F}}^{\mathrm{v}}$, the virtual link $l_{\mathrm{CF}}^{\mathrm{s}}$ is embedded to the substrate link $l_{\mathrm{GF}}^{\mathrm{s}}$, and the virtual link $l_{\mathrm{ac}}^{\mathrm{v}}$ is embedded to the substrate link $l_{\mathrm{CF}}^{\mathrm{s}}$. At this time, the substrate network would have opened three nodes. When the virtual node $n_{\mathrm{c}}^{\mathrm{v}}$ is embedded to the substrate node $n_{\mathrm{A}}^{\mathrm{v}}$, the virtual link $l_{\mathrm{bc}}^{\mathrm{v}}$ is embedded to the substrate links $l_{\mathrm{GF}}^{\mathrm{s}}$ and $l_{\mathrm{FA}}^{\mathrm{s}}$, the virtual link $l_{\mathrm{ac}}^{\mathrm{v}}$ is embedded to the substrate link $l_{\mathrm{CA}}^{\mathrm{s}}$. At this time, the substrate network would have opened four nodes. Although the substrate nodes $n_{\mathrm{F}}^{\mathrm{v}}$ do not carry virtual nodes, they are in an open state. These nodes are called host nodes. Compared with the case of the embedding virtual nodes $n_{\mathrm{c}}^{\mathrm{v}}$ to the substrate nodes $n_{\mathrm{F}}^{\mathrm{v}}$, embedding the virtual nodes $n_{\mathrm{c}}^{\mathrm{v}}$ to the substrate nodes $n_{\mathrm{A}}^{\mathrm{v}}$ consumes more energy.

Case 2. Link congestion problem affects the embedding performance.

When the virtual node $n_{\mathrm{c}}^{\mathrm{v}}$ is embedded to the substrate node $n_{\mathrm{B}}^{\mathrm{v}}$, the virtual link $l_{\mathrm{bc}}^{\mathrm{v}}$ is embedded to the substrate link $l_{\mathrm{GC}}^{\mathrm{s}}$ and the link $l_{\mathrm{CB}}^{\mathrm{s}}$, and the virtual link $l_{\mathrm{ac}}^{\mathrm{v}}$ is embedded to the substrate link $l_{\mathrm{CB}}^{\mathrm{s}}$. At this time, the substrate network would have opened three nodes. However, due to the insufficient bandwidth, the substrate link $l_{\mathrm{CB}}^{\mathrm{s}}$ is not enough to carry the virtual links $l_{\mathrm{bc}}^{\mathrm{v}}$ and $l_{\mathrm{ac}}^{\mathrm{v}}$. Therefore, the second embedding scheme can be selected; that is, the virtual link $l_{\mathrm{bc}}^{\mathrm{v}}$ is embedded to the substrate links $l_{\mathrm{GF}}^{\mathrm{s}}$, $l_{\mathrm{FA}}^{\mathrm{s}}$, and $l_{\mathrm{AB}}^{\mathrm{s}}$, and the virtual link $l_{\mathrm{ac}}^{\mathrm{v}}$ is embedded to the substrate link $l_{\mathrm{CB}}^{\mathrm{s}}$. At this time, the substrate link carrying the virtual link is too long, and it consumes more energy. Moreover, the bandwidth of the substrate link $l_{\mathrm{AB}}^{\mathrm{s}}$ cannot satisfy the bandwidth request of the virtual link $l_{\mathrm{bc}}^{\mathrm{v}}$. Therefore, when the distance between the substrate nodes is extremely long, the substrate link will be considerably long as well, which can easily cause link congestion problems and increase the embedding failure rate.

Case 3. Resource richness of the substrate nodes affects the embedding performance.

When the virtual node $n_{\mathrm{c}}^{\mathrm{v}}$ is embedded to the substrate node $n_{\mathrm{H}}^{\mathrm{v}}$, the virtual link $l_{\mathrm{bc}}^{\mathrm{v}}$ is embedded to the substrate link $l_{\mathrm{GH}}^{\mathrm{s}}$, and the virtual link is $l_{\mathrm{ac}}^{\mathrm{v}}$ embedded to the substrate link $l_{\mathrm{CH}}^{\mathrm{s}}$. At this time, the substrate network would have opened three nodes. When comparing the embedding of node $n_{\mathrm{c}}^{\mathrm{v}}$ to $n_{\mathrm{F}}^{\mathrm{v}}$, although both of the schemes would have opened three substrate nodes, the resource richness of node $n_{\mathrm{H}}^{\mathrm{v}}$ is lower

than that of node n_F^v. Although this scheme can meet the current embedding requirements, when new embedding occurs, it is difficult to meet the embedding requirements without opening new nodes. For example, when considering virtual network 2 (vn2), if virtual node n_c^v is embedded to the substrate node n_F^v in the previous step, the requirements for carrying vn2 can still be met in the opened nodes and links (we can embed the virtual node n_d^v to the substrate node n_G^v, embeds the virtual node n_e^v to substrate noden_F^v, and embeds the virtual link l_{de}^v to the substrate link l_{GF}^s). However, if the virtual node n_c^v is embedded to the substrate node n_H^v in the previous step, an embedding scheme that can embed vn2 cannot be found in the area where the nodes and links are currently turned on; hence, the new nodes and links must be reopened.

Case 4. Substrate node connection density affects the embedding performance.

The connection density between the substrate nodes also reflects the overall performance of the virtual network mapping to a certain extent. For example, consider the case when the virtual node n_c^v is embedded to the substrate noden_F^v. As the connection density of the node n_F^v is higher, when a new virtual network occurs, apart from the link directly connected to the node, which can be used as the embedding candidate set, the substrate links l_{FE}^s and l_{EA}^s can also be embedded. Therefore, when the virtual nodes are embedded to n_F^v, the link embedding success rate is higher than that of substrate nodes with a lower connection density.

Based on the above analysis, it can be observed that the structural characteristics of the nodes are crucial to the virtual network embedding performance. We can summarize the following points. (1) Enabling the host nodes requires energy consumption, and reducing the number of host nodes will help reduce the energy consumption. (2) The link congestion problem will reduce the acceptance ratio. Therefore, the short link should be selected during embedding. (3) Choosing the nodes with abundant substrate resources helps increase the success rate of the subsequent virtual network embedding. (4) Selecting substrate nodes with a high connection density is beneficial to increase the success rate of the subsequent virtual network embedding. From this, the wavelet diffusion method is used to comprehensively consider the structural characteristics of the node and it initializes the substrate network. Subsequently, it selects the resource-rich area in the substrate network according to the structural characteristics of the node, and it defines this area as the initialization area, wherein we apply the Hopfield network for virtual network embedding.

3.1.2. Calculate the Structural Feature Value of the Node Structure. Wavelet diffusion is a multiscale dimensionality reduction analysis framework, which effectively describes the multiscale characteristics of the data through the diffusion operator. Through the analysis of the structural characteristics of the nodes, this study considers the number of host nodes, the link bandwidth, the substrate resource richness, and the node connection density. This is achieved by using the wavelet diffusion principle and defining the energy diffusion equation. By analyzing the structure of the node, generating the structural feature value of the node, and using this as the standard to select the node candidate set, we can select the appropriate embedding range for the subsequent operation of the Hopfield neural network.

We modeled the substrate network as a weighted undirected graph, denoted as $G_S = (N^S, L^S)$, where N^s and L^s represent the substrate nodes and substrate links, respectively. If there is a link between the substrate node $n_i^s \in N^S$ and the substrate node $n_i^s \in N^S$, we can define the weight A_{ij} of the link as shown in

$$A_{ij} = \begin{cases} \mathrm{ReBW}\left(l_j^s\right), & \text{if } l_j^s \in L^s, \\ 0, & \text{else}, \end{cases} \tag{10}$$

where Re BW(l_j^s) represents the remaining bandwidth of the link. The diffusion distance between any two nodes in the substrate network node is defined as $d(i, j)$ for $\forall n_i^s \in N^s, \forall n_j^s \in N^s$. $d(i, j) \geq 0$ is satisfied, if and only if $i = j$ and $d(i, j) = 0$. The link weight A_{ij} is regarded as a diffusion operator and $D(ij) = \sum_{j=}^{n} A_{ij}$. Assuming that the substrate node is a diffusion source, the energy intensity of the diffusion source is defined as $\phi = (\phi_1, \phi_2 \cdots \phi_n)$. According to Newton's law of cooling, the energy diffusion equation is defined by applying

$$\frac{d\phi_i}{dt} = -K\mathrm{A}(ij)\left(\phi_i - \phi_j\right), \tag{11}$$

where K is the diffusion coefficient, and the above equation is converted into a matrix form as shown below.

$$\frac{d\phi}{dt} = -K\mathrm{L}\phi, \tag{12}$$

where L is the Laplace matrix; if A is the adjacency matrix of G_S and D is the degree matrix of G_S, then L = D − A = $U\Delta U^T$, where U is the unit orthogonal eigenvector of L and h_i is the eigenvalue of L, then Δ = diag $(h_1, h_2, \cdots, h_n)$. The above formula is solved as shown below.

$$\begin{aligned} \phi(t) &= \phi(0) \cdot \varphi, \\ \varphi &= \mathrm{U}\begin{bmatrix} e^{-Kh_0 t} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & e^{-Kh_n t} \end{bmatrix}\mathrm{U}^{\mathrm{T}}, \end{aligned} \tag{13}$$

where t is the diffusion time, $\phi(0)$ is the energy value of G_S at $t = 0$, and the energy intensity at t is obtained by performing a calculation with φ. Then, the structural feature value (SFV) of the node n_i^s is defined as shown in

$$\mathrm{SFV} = \frac{1}{n-1}\sum_{j=1, j\neq i}^{n} \mathrm{SF}_{ij}, \tag{14}$$

where $\mathrm{SF}_{ij} = [\varphi_{i1}, \varphi_{i2}, \cdots, \varphi_{\mathrm{in}}]$ indicates that node n_i^{s} diffuses the energy value to the other substrate nodes. Notably, setting $j \neq i$ implies that the influence of the resource amount of the node itself on the energy value is not considered, which avoids the island node with the higher resources from obtaining a high evaluation value.

By applying the wavelet diffusion considering the structural features of the nodes of the substrate network, we can select the areas with richer resources in the substrate network and embed them in the areas.

3.1.3. Select the Candidate Area. The candidate area contains a set of nodes with richer resources, and the selection of this area is conducive to the subsequent operation of the Hopfield neural network. First, we can calculate the $\mathrm{SFV}^{\mathrm{v}}$ of the virtual node that is to be embedded, and we can calculate the maximum structural feature value $\mathrm{SFV}^{\mathrm{v}}_{\max}$. Then, we calculate the structural feature value $\mathrm{SFV}^{\mathrm{s}}$ of the substrate network node and determine the initial candidate area as shown in

$$\mathrm{CA}_{\mathrm{init}} = \{v_i\}, \tag{15}$$

where $v_i \in \arg\operatorname{sort}(\mathrm{SFV}^{\mathrm{s}}_{\mathrm{c}})$ and $\mathrm{SFV}^{\mathrm{s}}_{\mathrm{c}} \geq \mathrm{SFV}^{\mathrm{v}}_{\max}$. In other words, the SFV set of the substrate nodes in the substrate network is greater than or equal to the SFV of the virtual nodes that are selected as the initial candidate area. If the embedding path cannot be found in the initial area, the number of virtual network nodes is used as the expansion size, and the candidate areas are sequentially expanded in the descending order of the $\mathrm{SFV}^{\mathrm{s}}$ values until the embedding is successful.

3.2. ALM-Based Problem Conversion Process. The energy-efficient VNE problem includes the equality constraints and inequality constraints. Under the original model, we could not derive $f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}}$. Hence, we converted the original VNE problem into an unconstrained problem, which includes $f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}}$ in the Lagrange function to facilitate the subsequent processing of the Hopfield neural network. According to the proposed energy consumption model of the VNE, discussed in Section 2, the energy consumption of the link is constant, and most of the energy consumption comes from the active substrate nodes (the substrate node that hosted a virtual node). Therefore, reducing the energy consumption of the substrate nodes can reduce the overall embedding energy consumption. According to Equations (3)–(9), the optimization problem of the virtual network energy consumption is simplified as follows:

$$\min \sum_{n_i^{\mathrm{s}} \in N^{\mathrm{S}}} \mathrm{EN}_i, \tag{16}$$

Constraints:

$$\sum_{n_i^{\mathrm{s}} \in N^{\mathrm{S}}} \mathrm{ReqCPU}(n_i^{\mathrm{s}}) - \sum_{n_i^{\mathrm{v}} \in N^{\mathrm{V}}} \mathrm{ReqCPU}(n_i^{\mathrm{v}}) \geq 0, \tag{17}$$

$$\sum_{n_i^{\mathrm{v}} \in N^{\mathrm{V}}, n_i^{\mathrm{s}} \in N^{\mathrm{S}}} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}} - 1 = 0, \tag{18}$$

$$1 - \sum_{n_i^{\mathrm{v}} \in N^{\mathrm{V}}, n_i^{\mathrm{s}} \in N^{\mathrm{S}}} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}} \geq 0. \tag{19}$$

It is complicated to directly solve the aforementioned problems. Therefore, the augmented Lagrangian multiplier method is used to convert the abovementioned problems with constraints into unconstrained problems. The augmented Lagrangian function is constructed as follows:

$$\begin{aligned}\varphi\left(f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}}, \lambda, \mu, \sigma\right) = {} & \sum_{n_i^{\mathrm{s}} \in N^{\mathrm{S}}} \mathrm{EN}_i + \frac{1}{2\sigma} \sum_{i=1}^{m} \left\{ \left[\max\left(0, \lambda_i - \sigma\left(\sum_{n_i^{\mathrm{s}} \in N^{\mathrm{S}}} \mathrm{ReCPU}(n_i^{\mathrm{s}}) - \sum_{n_i^{\mathrm{v}} \in N^{\mathrm{V}}} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}} \cdot \mathrm{ReqCPU}(n_i^{\mathrm{v}})\right)\right)\right]^2 - \lambda_i^2 \right\} \\ & + \sum_{j=1}^{l} \mu_i \left(\sum_{n_i^{\mathrm{v}} \in N^{\mathrm{V}}, n_i^{\mathrm{s}} \in N^{\mathrm{S}}} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}} - 1 \right) + \frac{\sigma}{2} \sum_{j=1}^{l} \mu_i \left(\sum_{n_i^{\mathrm{v}} \in N^{\mathrm{V}}, n_i^{\mathrm{s}} \in N^{\mathrm{S}}} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}} - 1 \right)^2 \\ & + \frac{1}{2\sigma} \sum_{i=1}^{m} \left\{ \left[\max\left(0, \lambda_j - \sigma\left(1 - \sum_{n_i^{\mathrm{s}} \in N^{\mathrm{S}}, n_i^{\mathrm{v}} \in N^{\mathrm{V}}} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}}\right)\right)\right]^2 - \lambda_j^2 \right\},\end{aligned} \tag{20}$$

where λ and μ are Lagrange multipliers of the inequality constraints and the equality constraints, respectively, and σ is a penalty factor. Therefore, the unconstrained problem of the VNE is described as follows:

$$\min \varphi\left(f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}}, \lambda, \mu, \sigma\right). \tag{21}$$

The iterative equations of the Lagrange multipliers are presented below:

$$\lambda_i^{(l+1)} = \max\left(0, \lambda_i^{(l)} - \sigma\left(\sum_{n_i^{\mathrm{s}} \in N^{\mathrm{S}}} \mathrm{ReCPU}(n_i^{\mathrm{s}}) - \sum_{n_i^{\mathrm{v}} \in N^{\mathrm{V}}} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}(l)} \cdot \mathrm{ReqCPU}(n_i^{\mathrm{v}})\right)\right), \tag{22}$$

$$\lambda_j^{(l+1)} = \max\left(0, \lambda_j^{(l)} - \sigma\left(1 - \sum_{n_i^{\mathrm{s}} \in N^{\mathrm{S}}, n_i^{\mathrm{v}} \in N^{\mathrm{V}}} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}(l)}\right)\right), \tag{23}$$

$$\mu_j^{(l+1)} = \mu_i^{(l)} + \sigma \sum_{n_i^{\mathrm{v}} \in N^{\mathrm{V}}, n_i^{\mathrm{s}} \in N^{\mathrm{S}}} f_{n_i^{\mathrm{s}}}^{n_i^{\mathrm{v}}(l)} - 1. \tag{24}$$

Through the above transformation, adding the constraint conditions of the VNE problem to the unconstrained function can avoid the disadvantages of the traditional penalty function method. Since the penalty function is relatively sensitive to the choice of the initial value, it usually tends to be infinity, which causes the augmented objective function to deviate from the initial solution direction. This condition causes the saturated network to interrupt or ignore the restriction of the constraints. In the end, only an approximate solution is obtained. The augmented Lagrangian multiplier is introduced to the penalty term of the penalty function method to avoid increasing the penalty parameter to infinity

and removing it from the limit of the initial value. By doing this, the algorithm successfully converges to the optimal solution when the constraints are met and an initial arbitrary value is set.

3.3. Hopfield Neural Network-Based Problem Solving. We used the Hopfield neural network to solve the above ALM problem and the neuron state to represent the binary variable $f_{n_i^s}^{n_i^v}$. The binary variable represents the virtual node embedding, which includes the embedding path of the virtual node. Therefore, the solving process of the binary variables involves solving the virtual network node embedding problem. With the unconstrained problem (20) that serves as the energy function of the Hopfield neural network, the entire network tends to be balanced by solving the energy function to obtain the network output state. The energy function is shown in Equation (18).

$$E = \varphi\left(f_{n_i^s}^{n_i^v}, \lambda, \mu, \sigma\right). \tag{25}$$

The dynamic equation of the Hopfield neural network for $f_{n_i^s}^{n_i^v}$ is constructed by the energy function, and its input state is shown as follows:

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} - \frac{\partial E}{\partial\left(f_{n_i^s}^{n_i^v}\right)}. \tag{26}$$

The state is updated according to the first-order Euler method as shown in Equation (20).

$$du_i(t+1) = u_i(t) + \frac{du_i}{dt}\Delta t. \tag{27}$$

The activation function of the network is expressed as follows:

$$f_{n_i^s}^{n_i^v} = g(u_i(t)) = \frac{1}{2}\left(1 + \tan\operatorname{sig}\left(\frac{u_i(t)}{u_0}\right)\right). \tag{28}$$

With ReqCPU(n_i^v), CPU(n_i^s), and ReCPU(n_i^s) as the inputs, the Lagrangian multipliers λ_i, λ_j, and μ_i are updated by Equations (22), (23), and (24), respectively. The binary variable of the virtual network node embedding $f_{n_i^s}^{n_i^v}$ is iterated sequentially on the basis of Equations (25)–(28) to generate the VNE data flow as follows:

$$\begin{aligned}&\mathrm{ReqCPU}(n_i^v), \mathrm{CPU}(n_i^s), \mathrm{ReqCPU}(n_i^s) \longrightarrow f_{n_i^s}^{n_i^v(0)} \longrightarrow \lambda_i^{(0)}, \lambda_j^{(0)}, \mu_i^{(0)} \\ &\longrightarrow \cdots \longrightarrow f_{n_i^s}^{n_i^v(l)} \longrightarrow \lambda_i^{(l)}, \lambda_j^{(l)}, \mu_i^{(l)} \longrightarrow \cdots \longrightarrow f_{n_i^s}^{n_i^v(k)}.\end{aligned} \tag{29}$$

We constructed a Hopfield neural network and initialized the network parameters. Then, we extracted the values of ReqCPU(n_i^v), CPU(n_i^s), and ReCPU(n_i^s) on the basis of the virtual network state and set the threshold ε and the maximum number of iterations θ. When $|f_{n_i^s}^{n_i^v(l+1)} - f_{n_i^s}^{n_i^v(l)}| \le \varepsilon$ or when the algorithm reaches the maximum iteration number θ, the value of $f_{n_i^s}^{n_i^v}$ is obtained; this value represents the virtual node embedding scheme. The main process is presented in Algorithm 1.

The binary variable matrix of the virtual network node that embeds $f_{n_i^s}^{n_i^v}$ is obtained by Algorithm 1, and the virtual node embedding scheme can be determined. Then, the shortest path method is used for virtual link embedding to obtain the energy-efficient VNE scheme.

4. Performance Evaluation

4.1. Simulation Environment. To evaluate the performance of the proposed algorithm, two classical simulation environments were developed, namely, a medium-sized scenario [7, 23] and small-sized scenario [24].

In the medium-sized scenario, the substrate network consists of 100 substrate nodes, which are randomly connected with a probability of 0.5. The substrate node's CPU and the link's bandwidth capacities obey a uniform distribution from 50 units to 100 units. The number of virtual nodes in the virtual request obeys the uniform distribution between two and 10. Each pair of virtual nodes is randomly connected with a probability of 0.5. The CPU of the virtual nodes and the bandwidth of the virtual requirements are uniformly distributed between 0 and 20 units.

In the small-sized scenario, the substrate network consists of 50 substrate nodes, which are randomly connected with a probability of 0.5. The substrate node's CPU and the link's bandwidth capacities follow a uniform distribution from 50 units to 100 units. The number of virtual nodes in the virtual request obeys the uniform distribution between two and 10. Each pair of virtual nodes is randomly connected with a probability of 0.5. The CPU and bandwidth resources of the virtual nodes and links obey the uniform distribution of [0, 10] and [0, 20], respectively. In both configurations, the arrival of the virtual requests obeys the Poisson process, and its lifetime follows an exponential distribution. The substrate and virtual network topologies are generated by the GT-ITM tool [25]. We fixed the virtual network's request arrival rate to four per 100 time units in the medium-sized scenario and to five per 100 time units in the small-sized scenario. The average virtual network lifetime was set to 1000 time units. Each simulation is run for 10,000 time units.

4.2. Performance Evaluation Metrics. In this study, the ELECTRE_VNE [26], EE_CTA [27], and AEF algorithms [28], which are energy-efficient algorithms, were selected for a comparative study. The following metrics are used to demonstrate the energy-saving effect of the proposed algorithm.

The average acceptance ratio (AAR) can be determined as follows:

$$\mathrm{AAR} = \frac{A_T}{C_T}, \tag{30}$$

Input: ReqCPU(n_i^v), CPU(n_i^s), ReCPU(n_i^s)
Output: $f_{n_i^s}^{n_i^v}$
(1) Network initialization, parameter selection.
(2) Calculate the ALM function according to Equations (21)–(24), and assign it to the neural network energy function E.
(3) Calculate du_i/dt according to the dynamic function in Equation (26), and iteratively update it according to Equation (27).
(4) Calculate the binary variable $f_{n_i^s}^{n_i^v}$ according to the activation function in Equation (28).
(5) Determine if the iteration is over. If iter > 1000, then the algorithm is terminated. Otherwise, iter = iter + 1, and the algorithm returns to step 2.

Algorithm 1:

where C_T represents the total number of virtual network requests in the time period from 0 to T.

The average number of open nodes (ANON) is expressed as follows:

$$\mathrm{ANON} = \frac{\sum_{i=1}^{N_T} \mathrm{NO}_i}{N_T}, \tag{31}$$

where N_T indicates the number of all the valid time periods from 0 to T. NO_i indicates the number of substrate nodes that are active at time period i.

The average number of open links (ANOL) can be identified with the following expression:

$$\mathrm{ANOL} = \frac{\sum_{i=1}^{N_T} \mathrm{LO}_i}{N_T}, \tag{32}$$

where LO_i represents the number of substrate links that are active at time period i.

The average amount of the energy consumption (AAEC) can be determined as follows:

$$\mathrm{AAEC} = \frac{\sum_{i=1}^{N_T} \left(\sum_{n_i^v \in N^V, n_i^s \in N^S} \mathrm{EN}_i + \sum_{l_j^v \in L^V, l_j^s \in L^S} \mathrm{EN}_j \right)}{N_T}, \tag{33}$$

where $\sum_{n_i^v \in N^V, n_i^s \in N^S} \mathrm{EN}_i + \sum_{l_j^v \in L^V, l_j^s \in L^S} \mathrm{EN}_j$ is the energy consumption of the substrate network at the i time period.

The average ratio of the revenue and cost (ARRC) can be calculated by applying the following expression:

$$\mathrm{ARRC} = \frac{\sum_{i=1}^{A_T} \mathrm{Revenue}_{R_i}}{\sum_{i=1}^{A_T} \mathrm{Cost}_{R_i}}. \tag{34}$$

From this, A_T represents the number of virtual requests that were successfully accepted in the time period from 0 to T. The parameters $\mathrm{Revenue}_{R_i}$ and Cost_{R_i} represent the benefits and costs of the mapped virtual network

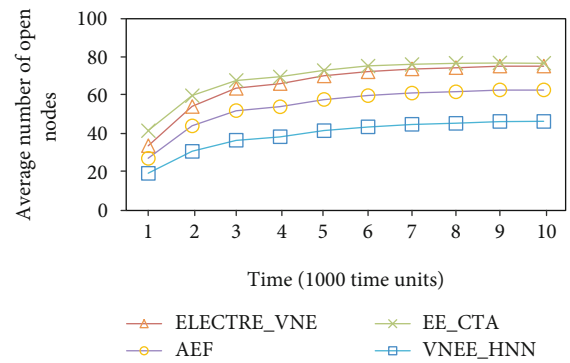

Figure 2: Average number of open nodes (medium-sized scenario).

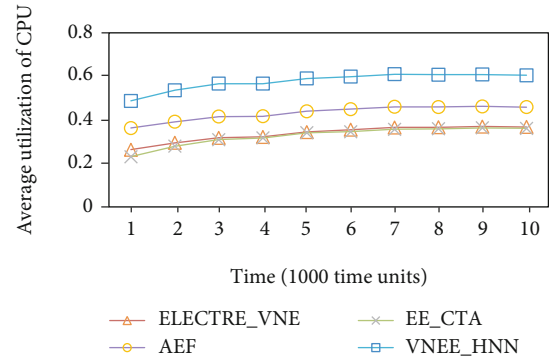

Figure 3: Average utilization of CPU (medium-sized scenario).

R_i, respectively. The methods to calculate these parameters are as follows:

$$\mathrm{Revenue}_{R_i} = \sum_{n_i^v \in N^V} \mathrm{AccCPU}(n_i^v) + \sum_{l_j^v \in L^V} \mathrm{AccBW}\left(l_j^v\right),$$

$$\mathrm{Cost}_{R_i} = \sum_{n_i^v \in N^V, n_i^s \in N^S} f_{n_i^s}^{n_i^v} \cdot \mathrm{AccCPU}(n_i^v) + \sum_{l_j^v \in L^V, l_j^s \in L^S} f_{l_j^s}^{l_j^v} \cdot \boldsymbol{AccBW}\left(l_j^v\right), \tag{35}$$

where $\boldsymbol{AccCPU}(n_i^v)$ is the virtual node resource that has been successfully mapped and $\mathrm{AccBW}(l_j^v)$ is the virtual link resource that has been successfully mapped. The virtual nodes belonging to the same virtual network are not

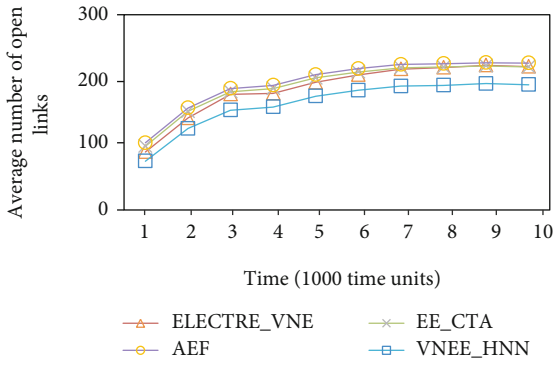

Figure 4: Average number of open links (medium-sized scenario).

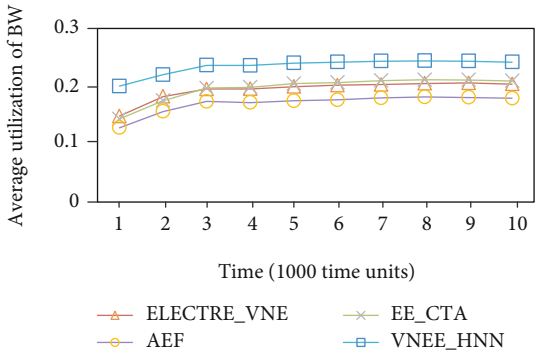

Figure 5: Average utilization of BW (medium-sized scenario).

allowed to be embedded in the same substrate node. In contrast, the virtual links of the same virtual network are allowed to be embedded in the same substrate link. Therefore, the ARRC is related to the embedded length of the substrate path. The shorter the substrate path is, the higher the ARRC is.

The ratio of the node load balancing (RNLB) can be determined as follows:

$$\boldsymbol{RNLB} = \sqrt{\frac{1}{\sum_{i=1}^{N_T} NO_i} \sum_{n_i^s \in N^s} \left(u^{n_i^s} - \overline{u^{n_i^s}}\right)^2}, \tag{36}$$

where $\overline{u^{n_i^s}}$ is the average value of the node resource utilization.

The ratio of the link load balancing (RLLB) can be calculated by applying the following expression:

$$\boldsymbol{RLLB} = \sqrt{\frac{1}{\sum_{i=1}^{N_T} LO_i} \sum_{l_j^s \in L^s} \left(u^{l_j^s} - \overline{u^{l_j^s}}\right)^2}, \tag{37}$$

where $(u^{l_j^s} = \boldsymbol{ReqBW}(l_j^s)/\boldsymbol{BW}(l_j^s))$ represents the bandwidth resource utilization of the link l_j^s. $\boldsymbol{ReqBW}(l_j^s)$ is the request resource of the virtual link l_j^s, and $\mathrm{BW}(l_j^s)$ represents the total resources of the substrate link l_j^s. $\overline{u^{l_j^s}}$ is the average utilization of the link resource.

4.3. Results and Discussion

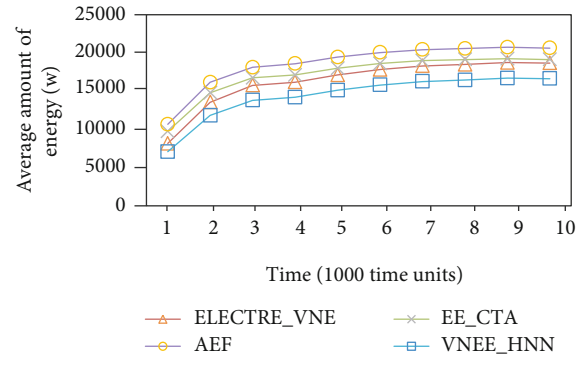

Figure 6: Average amount of energy (w) (medium-sized scenario).

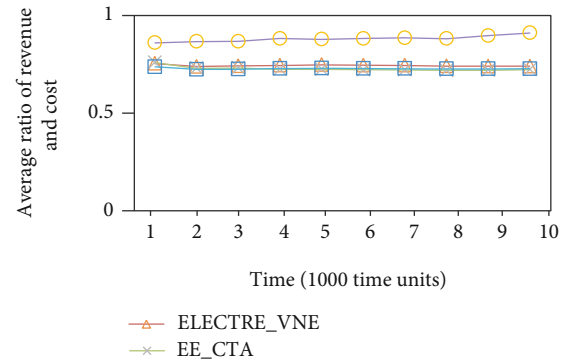

Figure 7: Average ratio of revenue and cost (medium-sized scenario).

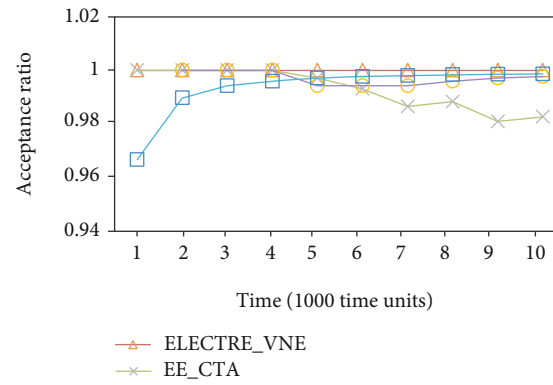

Figure 8: Acceptance ratio (medium-sized scenario).

4.3.1. Medium-Sized Scenario. Figures 2–8 show the performance in the medium-sized scenario. Figure 2 shows the average number of the open nodes in the medium-sized scenario. The number of nodes that are opened by the VNEE_HNN algorithm is less than the other three algorithms. After running this for 10,000 time units, the number of open nodes remained at 46. The ELECTRE_VNE, EE_CTA, and AEF algorithms enable 75, 77, and 63 substrate nodes, respectively. This shows that we can use the wavelet diffusion to provide candidate sets for the virtual network embedding, eliminate a batch of inappropriate node sets, and use the Hopfield neural network to find the embedding solutions in resource-rich areas, which reduces the number of open nodes

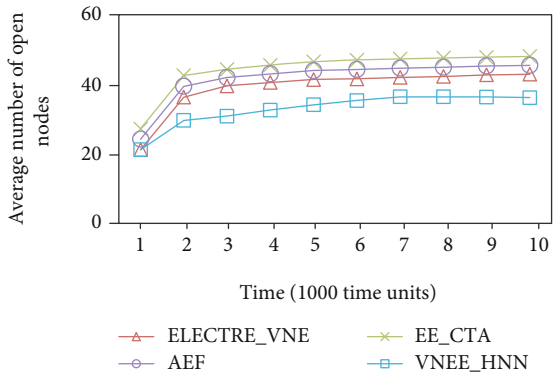

Figure 9: Average number of open nodes (small-sized scenario).

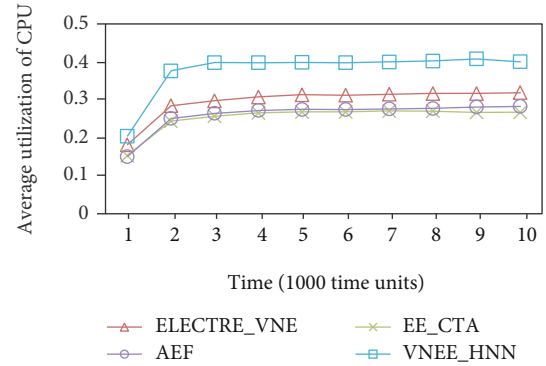

Figure 10: Average utilization of CPU (small-sized scenario).

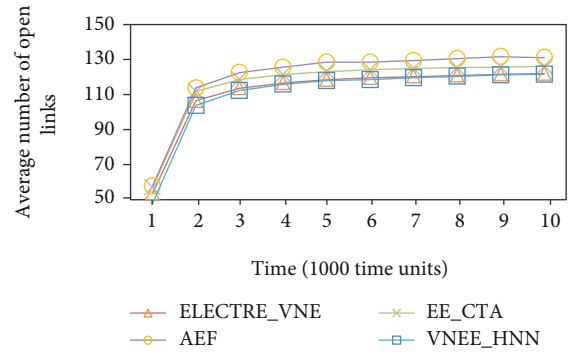

Figure 11: Average number of open links (small-sized scenario).

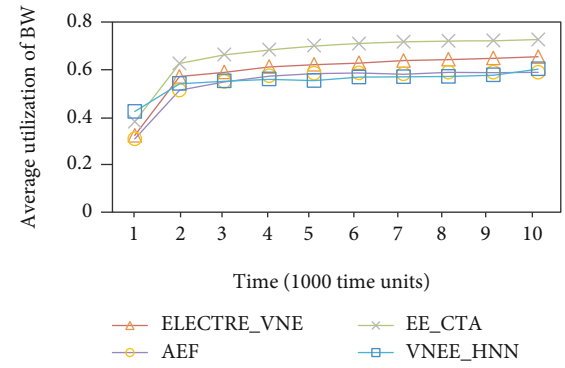

Figure 12: Average utilization of BW (small-sized scenario).

to a certain extent. Figure 3 shows the average use of the CPU. After running it for 10,000 time units, the node resource utilization of the VNEE_HNN algorithm was at about 60%, which is higher than that of the ELECTRE_VNE (37%), EE_CTA (36%), and AEF algorithms (46%). This is because the VNEE_HNN algorithm is different from the other direct embedding algorithms. Before the embedding stage, the structural information of the node can find the embedding area with a greater embedding potential; hence, the opened nodes have a greater probability of accepting the subsequent virtual nodes. Therefore, the node resource utilization rate improves. Figure 4 shows the average number of open links. After running for 10,000 time units, the link opening amount of the VNEE_HNN algorithm is at 192. The corresponding values with ELECTRE_VNE, EE_CTA, and AEF algorithms are 220, 219, and 226, respectively. The VNEE_HNN algorithm only maintains a weak advantage in the link performance. This is because the position of the node is determined first, and there are certain limitations to the embedding position of the subsequent link, which leads to a reduction in the selectable path. Figure 5 shows the average utilization of the BW. It can be observed from the figure that the resource utilization rate of the VNEE_HNN algorithm is about 24%, which is lower than the corresponding values of ELECTRE_VNE, EE_CTA, and AEF algorithms by 4%, 3%, and 6%, respectively. This is due to the limitation of the link locations, which reduces the number of substrate links that repeatedly carries the virtual links; thus, reducing the link bandwidth resource utilization. Figure 6 shows the average amount of energy for each algorithm. From Figure 6, we can see that the ELECTRE_VNE, EE_CTA, AEF, and the VNEE_HNN algorithms consume approximately 18,680 w, 19,158 w, 20,642 w, and 16,649 w, respectively, after running for 10,000 time units. It can be observed that the VNEE_HNN algorithm has a low energy consumption. This is because the algorithm increases the number of repeated embeddings of the nodes. This reduces the total number of nodes that are opened, thereby reducing the total energy consumption. Figure 7 presents the average ratio of the revenue and cost. It can be observed that the AEF algorithm is the highest, which is maintained at about 90% after running 10,000 time units. At the same time, the other three algorithms remain at about 73%. This is because the AEF algorithm has a high amount of link openings and a low link resource utilization. A trade-off is made between the embedding load and the cost. The virtual links are mapped to the substrate links in a balanced manner as much as possible. As a result, this improves the performance of the links and the average ratio of the revenue and cost. Figure 8 displays the acceptance ratio. It can be observed that there is a little difference in the acceptance rate of the four algorithms, which remains at approximately 99%.

4.3.2. Small-Sized Scenario. Figures 9–15 show the experimental results of the algorithms in the small-sized scenario. Figure 9 shows the average number of open nodes under this scenario. It was demonstrated that the ELECTRE_VNE, EE_CTA, AEF, and the VNEE_HNN algorithms enabled 36, 43, 48, and 45 nodes, respectively, after running for 10,000 time

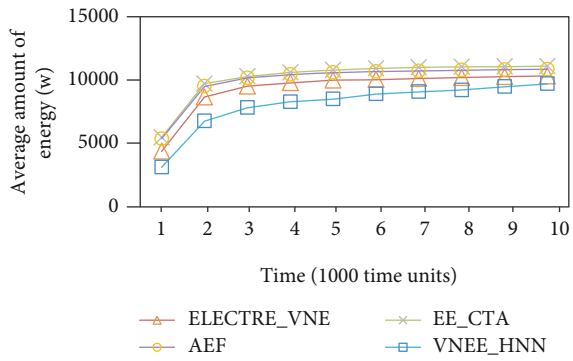

Figure 13: Average amount of energy (w) (small-sized scenario).

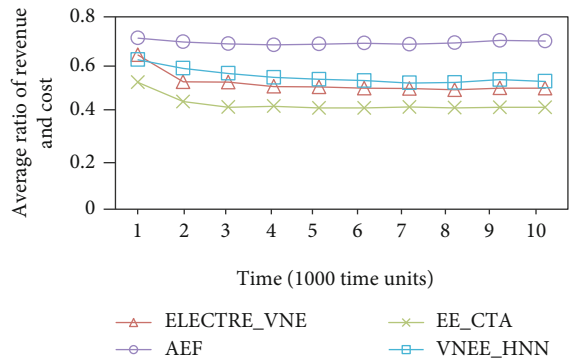

Figure 14: Average ratio of revenue and cost (small-sized scenario).

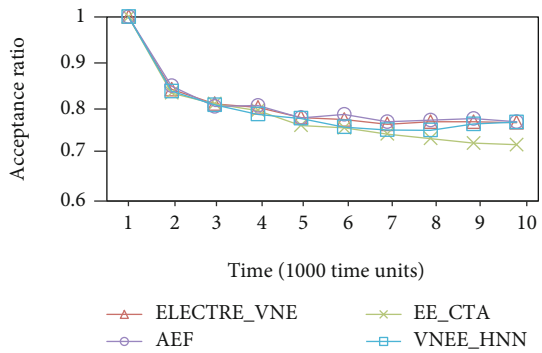

Figure 15: Acceptance ratio (small-sized scenario).

units. The VNEE_HNN algorithm maintains a small advantage since the substrate resources are not rich in comparison to the medium-sized scenario. This results in the preprocessing operation, which does not filter enough substrate nodes for the virtual network. Figure 10 depicts the average utilization of the CPU and the ELECTRE_VNE, EE_CTA, AEF, and VNEE_HNN algorithms after running for 10,000 time units, which are, respectively, 32%, 27%, 28%, and 36%. The performance is generally not high; however, the VNEE_HNN algorithm still maintains its advantages and reuses the opened nodes as much as possible with the lower node opening amount. Figure 11 displays the average number of open links. The opening amounts of the four algorithms remain at approximately 120. Figure 12 depicts the average utilization of the BW and the ELECTRE_VNE, EE_CTA, AEF, and the VNEE_HNN algorithms, which are about 65%, 73%, 59%, and 61% after running for 10,000 time units, respectively. Figure 13 shows the average amount of energy consumed. After running for 10,000 time units, the four algorithms were maintained at approximately 10,000 w. The VNEE_HNN algorithm has a slight advantage. By maintaining a lower number of active nodes, the total energy consumption of the VNEE_HNN algorithm reduced. Figure 14 shows the average ratio of the revenue and cost for the four algorithms. In the case of having insufficient resources, in addition to the AEF algorithm that still maintains a higher revenue-cost ratio, the revenue-cost ratios of the ELECTRE_VNE, EE_CTA, and VNEE_HNN algorithms are generally not high, and the ratios are maintained at approximately 50%. This is because the calculation of the revenue-cost ratio is related to the embedding length of the substrate path. The link resource utilization of the AEF algorithm is low, and the number of opening nodes is relatively large. Therefore, the embedding length is relatively long, and the matching degree between the substrate link and the virtual link is relatively high. They maintain a higher ratio of the revenue and cost. Figure 15 shows the results of the acceptance ratio for the four algorithms. The four algorithms remain at about 77% in the latter embedding stage.

5. Conclusion

In this study, the main optimization goal was to reduce the energy consumption in the VNE process. The Hopfield neural network was applied to the VNE problem. An energy-efficient embedding algorithm based on the Hopfield neural network was proposed. The proposed energy-efficient VNE constraint problem was converted into an unconstrained problem, and this problem was used as the energy function of the Hopfield neural network to obtain a virtual network node embedding scheme. Then, the shortest path algorithm was used to solve the link embedding scheme. To prove the effectiveness of the scheme, we created two classic experimental environments, namely, a medium-sized scenario and a small-sized scenario. The experiment results show that for the different scenarios, the algorithm can reduce the number of active substrate nodes and active substrate links while ensuring the AAR and AARC. The proposed algorithm is superior to other algorithms in terms of reducing the energy consumption, and it effectively solves the energy-efficient VNE problem. By establishing the model, this provides the basis for the subsequent problem-solving process. However, this work only considers the capacity constraints and the variable constraints. In our future work, we will focus on improving the energy-efficient model. Furthermore, we will consider adding other constraints, such as the location constraints and delays, which will improve the VNE model and obtain an increasingly effective embedding solution. In addition to the energy savings, there are other embedding properties in the embedding process, such as the acceptance ratio. There is a resource competition between them, and the trade-off between them will also be a focus in our future work.

Data Availability

The data used to support the findings of this study have not been made available because at this time as the data also forms part of an ongoing study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Funding

This work has been supported by State Grid Corporation of China science and technology project "Key technology and application of new multi-mode intelligent network for State Grid" (No. 5700-202024176A-0-0-00).

References

[1] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. de Turck, and R. Boutaba, "Network function virtualization: state-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] R. P. Esteves, L. Z. Granville, and R. Boutaba, "On the management of virtual networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 80–88, 2013.

[5] S. D. Qing, J. X. Liao, X. M. Zhu, J. Y. Wang, and Q. Qi, "Virtual network embedding algorithms in the network virtualization environment," *Journal of Software*, vol. 23, no. 11, pp. 3045–3058, 2012.

[6] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, vol. 1200no. 2006, pp. 1–12, Barcelona, Spain, 2006.

[7] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[8] X. Liu, B. Wang, S. Liu, Z. Yang, and Z. Zhao, "Heuristic algorithm for secure virtual network embedding," *Systems Engineering and Electronics*, vol. 40, no. 3, pp. 676–681, 2018.

[9] X. Liu, Z. Zhang, J. Li, and S. Su, "Clustering-based energy-aware virtual network embedding," *International Journal of Distributed Sensor Networks*, vol. 13, no. 8, 2017.

[10] M. He, L. Zhuang, S. Tian, G. Wang, and K. Zhang, "Multi-objective virtual network embedding algorithm based on q-learning and curiosity-driven," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, Article ID 150, 2018.

[11] L. Zhuang, S. Tian, M. He, G. Wang, W. Liu, and L. Ma, "Virtual network embedding algorithm via diffusion wavelet," *IEEE Access*, vol. 7, pp. 134145–134157, 2019.

[12] J. Wang, W. Chen, H. Cong, Z. Zhan, and J. Zhang, "An ant colony system based virtual network embedding algorithm," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1805–1810, Banff, AB, Canada, October 2017.

[13] H. K. Zheng, J.-J. Li, Y.-J. Gong et al., "Link mapping-oriented ant colony system for virtual network embedding," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1223–1230, San Sebastian, Spain, June 2017.

[14] Z. G. Chen, Z. H. Zhan, Y. Lin et al., "Multiobjective cloud workflow scheduling: a multiple populations ant colony system approach," *IEEE Transactions on Cybernetics*, vol. 49, no. 8, pp. 2912–2926, 2019.

[15] X. F. Liu, Z. H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 113–128, 2018.

[16] D. Chemodanov, F. Esposito, P. Calyam, and A. Sukhov, "A constrained shortest path scheme for virtual network service management," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 127–142, 2019.

[17] R. Mijumbi, J. Serrat, J. L. Gorricho, and R. Boutaba, "A path generation approach to embedding of virtual networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 334–348, 2015.

[18] Y. Wang, Q. Hu, and X. Cao, "A branch-and-price framework for optimal virtual network embedding," *Computer Networks*, vol. 94, pp. 318–326, 2016.

[19] Z. Yang and Y. Guo, "An exact virtual network embedding algorithm based on integer linear programming for virtual network request with location constraint," *China Communications*, vol. 13, no. 8, pp. 177–183, 2016.

[20] I. Houidi, W. Louati, and D. Zeghlache, "Exact multi-objective virtual network embedding in cloud environments," *The Computer Journal*, vol. 58, no. 3, pp. 403–415, 2015.

[21] V. Eramo, E. Miucci, and M. Ammar, "Study of migration policies in energy-aware virtual router networks," *IEEE Communications Letters*, vol. 18, no. 11, pp. 1919–1922, 2014.

[22] V. Eramo, E. Miucci, and M. Ammar, "Study of reconfiguration cost and energy aware VNE policies in cycle-stationary traffic scenarios," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1281–1297, 2016.

[23] X. Cheng, S. Su, Z. Zhang et al., "Virtual network embedding through topology awareness and optimization," *Computer Networks*, vol. 56, no. 6, pp. 1797–1813, 2012.

[24] X. L. Chang, X. M. Mi, and J. K. Muppala, "Performance evaluation of artificial intelligence algorithms for virtual network embedding," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 10, pp. 2540–2550, 2013.

[25] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of IEEE INFOCOM '96. Conference on Computer Communications*, vol. 1996, pp. 594–602, San Francisco, CA, USA, 1996.

[26] P. Zhang, H. Yao, C. Qiu, and Y. Liu, "Virtual network embedding using node multiple metrics based on simplified ELECTRE method," *IEEE Access*, vol. 6, pp. 37314–37327, 2018.

[27] A. Jahani, L. Khanli, M. Hagh, and M. A. Badamchizadeh, "EE-CTA: energy efficient, concurrent and topology-aware virtual network embedding as a multi-objective optimization problem," *Computer Standards & Interfaces*, vol. 66, pp. 103351–103368, 2019.

[28] D. Hu and Z. Yang, "An efficient and fast embedding algorithm for the virtual networks," *IEEE Access*, vol. 8, pp. 61528–61539, 2020.