Graph Convolutional Network Aided Virtual Network Embedding for Internet of Thing

Sihan Ma, Haipeng Yao, *Senior Member, IEEE*, Tianle Mai, *Student Member, IEEE*, Jingkai Yang, *Student Member, IEEE*, Wenji He, *Student Member, IEEE*, Kaipeng Xue, *Senior Member, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*—The past few years have seen the dramatic adoption of the Internet of Things (IoT) in everyday life, from manufacturing to healthcare. With the emergence of various new Internet of Things applications, it is a challenging problem to meet the different QoS requirements of Internet of Things applications in shared substrate networks. Recently, Network Virtualization (NV) has attracted a large amount of attention from academia and industry. NV enables multiple virtual networks to coexist on the same substrate network, thus providing IoT users with customized end-to-end services. The main challenge of NV is the Virtual Network Embedding (VNE) problem, which refers to embed different virtual networks into one substrate network. Inspired by the recent success of graph convolutional network (GCN) in graph structured data processing, in this paper, we propose a GCN aided VNE algorithm. The GCN can extract high-order spatial structure information among substrate nodes through the convolution kernel. Considering that the training data of VNE has no label, we introduce the policy gradient algorithm to optimize the GCN model. In addition, three evaluation metrics are designed to evaluate the performance of the network embedding policy. Some simulations are implemented to evaluate our proposed algorithm in comparison to the other state-of-the-art solutions.**

***Index Terms*—Graph convolutional network, policy gradient, reinforcement learning, virtual network embedding.**

I. Introduction

THE Internet of Things (IoT) refers to the billions of physical devices (e.g., automobiles, home appliances, mechanical arm) that are now connected to the Internet. This connectivity allows a higher degree of intelligence and automation by using Big Data technology to promote efficiency in performance and economic benefits. According to the Gartner's prediction, it is expected more than 25 billion IoT connections in the immediate future year 2025. However, with the emergence of new IoT applications (e.g., industrial IoT, high-definition video monitoring), the QoS requirements are becoming increasingly demanding. How to satisfy distinct QoS requirements of different IoT applications (e.g., time-sensitive applications, bandwidth-sensitive applications) in a shared substrate network becomes a challenging problem [1].

Recently, Network Virtualization (NV) has obtained a large amount of attention from both industry and academia [2], [3]. NV allows multiple virtual networks to coexist on a shared physical infrastructure [4]. By dynamically embedding different virtual networks into the physical network, the virtual networks can provide customized end-to-end guaranteed services to different IoT applications. For Network Virtualization, the most critical challenge is the Virtual Network Embedding (VNE) problem: efficiently mapping virtual nodes and links into a shared physical network, which is known to be NP-hard [5], [6]. In particular, the ever-increasing network complexity makes effective network embedding extremely difficult.

Inspired by machine learning, much of the current literature on VNE focuses specifically on deep learning-based solutions [7], [8]. Neural Networks (e.g., convolutional neural network, recurrent neural network) based VNE algorithms have gained massive success. However, traditional Neural Networks approaches could only be implemented using regular or Euclidean data. They can hardly extract the high-order adjoining relationships among nodes and links in VNE problems.

The non-regularity data structures have greatly promoted the development of Graph Neural Networks (GNN). GNN generalizes the neural network model to non-Euclidean graph-structured data. In the last few years, a variety of Graph Neural Networks have been developed, and Graph Convolutional Network (GCN) is one of them. The GCN can extract high-order spatial structure information among substrate nodes through the convolution kernel to represent the spatial structure of a graph [9]. Therefore, in this paper, we propose a GCN aided VNE scheme. Besides, due to the training data of the VNE does not have fixed labels, we use the policy gradient algorithm to optimize the parameters of the GCN model. Extensive and in-depth simulations are conducted to evaluate our proposed algorithm against other state of the art schemes.

Our paper makes the following two contributions:

Manuscript received 3 September 2020; revised 17 December 2021; accepted 6 September 2022. Date of publication 16 September 2022; date of current version 6 January 2023. This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1800805, in part by Artificial Intelligence and Smart City Joint Laboratory (BUPT-TGSTII) under Grant B2020001, in part by Future Intelligent Networking and Intelligent Transportation Joint Laboratory (BUPTCTTIC) under Grant B2019007, and in part by Intelligent Network Joint Laboratory (BUPTIN) under Grant B2021006. Recommended for acceptance by My Dr. T. Thai. *(Corresponding author: Haipeng Yao.)*

Sihan Ma, Haipeng Yao, Tianle Mai, Jingkai Yang, and Wenji He are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: 14404332617@qq.com; yaohaipeng@bupt.edu.cn; machealmai@gmail.com; yangjingkai@bupt.edu.cn; hewenji@bupt.edu.cn).

Kaipeng Xue is with the Department of Electronic Engineering, University of Science and Technology of China, Hefei 230052, China (e-mail: kpxue@ustc.edu.cn).

Mohsen Guizani is with the Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi 44737, UAE (e-mail: mguizani@gmail.com).

Digital Object Identifier 10.1109/TNSE.2022.3207205

- We apply GCN into the VNE problem. The multiple layer GCN can extract high-order spatial structure information among substrate nodes through the convolution kernel for end-to-end learning.
- We introduce the policy gradient algorithm to update the parameter of the policy network, which effectively improves the training performance.
- We design three evaluation metrics for the sate of evaluating the performance of VNE algorithms. Simulation results indicate that our GCN-VNE algorithm significantly outperformed to other baseline algorithms.

In Section II, we review some related work on the IoT virtualization solution considered. In Section III, we introduce our system model in detail. In Section IV, we propose a GCN-VNE algorithm. The implementation and performance of the GCN-VNE are described in Section V. Finally, our conclusion is presented in Section VI.

II. Related Work

In this part, we will give an introduction of the related work of virtual network embedding and graph convolutional networks.

A. Virtual Network Embedding

A large and growing body of literature has been investigated in the VNE problem. Generally, VNE is implemented in two phases, including node and link embedding phases. In some literature, these two stages are executed separately [10], [11], [12], [13], [14]. In [10], Yu et al. proposed a multipath based VNE algorithm. It allowed the substrate network to split and dynamically migrate the virtual link over multipath in the substrate network. The experiment result shown a significant improvement in the bandwidth resource utilization. Besides, to solve the limited resources problem in the VNE process, in [11], Razzaq et al. proposed a closest node-based VNE algorithm, which places the virtual vertices as closely as possible in the substrate and finds the closest paths for virtual edges that satisfy its demand. In [12], the Markov Random Walk model is applied by Cheng et al. to sort nodes according to the topological and resource properties. Then, the author proposed two algorithms to calculate the mapping priority based on the rank of the virtual node. In [13], Zhang et al. used node importance metric to rank virtual nodes in each VN request by base nodes according to node degree and clustering coefficient information.

In contrast, these two stages are executed together in some papers [15], [16], [17], [18]. In [15], Chowdhury et al. modeled the VNE problem as a mixed-integer program through the expansion of the substrate network. Besides, the author proposed the deterministic and randomized rounding algorithms for the network dynamical embedding. In [16], Lischka et al. proposed a subgraph isomorphism detection algorithm, which can complete node and link embedding in the same phase. The simulation results show that this algorithm is faster than the two-stages approaches.

However, as the network scale continues to expand, the heuristic algorithm suffers from the computation complexity problem. Recently, with the advancement of artificial intelligence, a multitude of researches on VNE architecture are carried out from the perspective of machine learning. In [19], Mijumbi et al. applied the deep neural network to the VNE problem. The experiment result demonstrated the good performance of the neural network based solution. In [20], Haeri et al. used the Monte-Carlo tree to search the optimal VNE solutions. Moreover, deep reinforcement learning (DRL) algorithms were introduced to VNE in [21], [22].

B. Graph Convolutional Network

Recently, graph neural networks have attracted increasing attention. GNN extends the deep neural network to graph-structured data, and is able to extract non-Euclidean graph-structured data information. There are mainly two types spatial feature extraction methods, the spatial domain and the spectral domain method. The spatial domain method extracts spatial features on topology graphs intuitively. In [23], a novel algorithm DeepWalk is proposed by Perozzi et al.. DeepWalk used local knowledge from the random walk algorithm as input to learn an encodes structural regularities representation. In [24], Jian et al. proposed a novel GCN algorithm, called LINE, where the first-order similarity and empirical probability of nodes are calculated respectively. The experiment data shown that the LINE performed well in any type of information network. In [25], Grover et al. learned the continuous feature representation of nodes when processing graph data, and proposed node2vec. In [26], Leonardo et al. extracted the potential representation of node structural identity and proposed the Struc2Vec algorithm. In [27], William et al. proposed an inductive framework, which can efficiently use other nodes to generate new node embedding vectors and is called GraphSAGE.

Accordingly, the spectral domain is to implement the convolution on the graph with the spectrum theory. To generalize the CNN to more general domains, in [28], Bruna et al. presented a hierarchical clustering based and spectrum of the graph Laplacian methods. Based on this, the author parameterized the graph convolution kernel and proposed the first-generation GCN formula. In [29], David et al. proposed a novel algorithm for performing Chebyshev polynomial approximation based wavelet transforms on arbitrary finite weighted large graphs. In [30], Thomas et al. presented a semi-supervised learning based algorithm on graph-structured data, which used parameter assumptions and re-regularization techniques to calculate the convolution kernel. In [31], Petar et al. proposed well-know GAN, which introduced the attention mechanism into the GCN and assigned the corresponding weights to different adjacent nodes.

III. System Model and Problem Formulation

In this section, the system model and problem formulation are introduced firstly. Then, we design three evaluation metrics to evaluate the performance of network embedding.

A. System Model

Let's assume a graph with N^s nodes and L^s links. Then, we abstract substrate networks into an undirected graph $G^S = (N^S, L^S, A_N^S, A_L^S)$, where A_N^S and A_L^S represent the resources capability of node N^s and link L^s. Similarly, we formulated the virtual network into $G^V = (N^V, L^V, C_N^V, C_L^V)$, where C_L^V and C_N^V are resource requirements sets of L^v and N^v. Therefore, the VNE problem can be formulated by $E : G^V(N^V, L^V) \rightarrow G^S(N, L)$, where $N \subset N^S$ and $L \subset L^S$. The node embedding process can be modeled by $X = \{x_{ij} \,|\, n_i \in N^V, n_j \in N^S\}$. When

$$\sum_{j=1}^{|N^S|} x_{ij} \;=\; 1, \tag{1}$$

it indicates that node n_i successfully embedded into the substrate node n_j. Similarly, the link embedded can be represented by $Y = \{y_{ij} \,|\, l_i \in L^V, l_j \in L^S\}$. When

$$\sum_{j=1}^{|L^S|} y_{ij} \;\geq\; 1, \tag{2}$$

it indicates that the link l_i is successfully embedded on one or more physical links.

Also, a virtual network requirement embedding needs to meet the constraints of link and node respectively. The restriction of node embedding can be expressed by:

$$x_{ij} \cdot C_{n_i}^V \leq x_{ij} \cdot A_{n_j}^S, \tag{3}$$

where $A_{n_j}^S$ indicates the remaining resources of n_j and $C_{n_i}^V$ represents the resource requirement of n_i. Also, the link restriction can be represented by:

$$y_{ij} \cdot C_{l_i}^V \leq y_{ij} \cdot A_{l_j}^S, \tag{4}$$

where $A_{l_j}^S$ denotes the remaining bandwidth of l_j and $C_{l_i}^V$ represents the demand of l_i.

For a clear understanding, an example is shown in Fig. 1. In Fig. 1, it consists of a substrate network and two requests 1 and 2. Request 1 requires a virtual network of 3 links and 3 nodes, and request 2 requires a virtual network of 4 links and 4 nodes. The numbers on the nodes and links denote their resource requirements. For the physical network, the numbers represent their remaining resources.

We assume that each physical node can support multiple virtual nodes. So a in request 1 and d in request 2 can be simultaneously embedded to the node A. Similarly, multiple virtual links can be mapped to the same physical link at the same time. The ab in request 1 and the de in request 2 can be simultaneously embedded to the link AB. We notice that the virtual link de in request 2 is embedded to the AB and BC, which cause extra bandwidth consumption. By observing the graph, we can easily find that move d in request 2 to the node B can save the bandwidth without affecting the embedding performance.

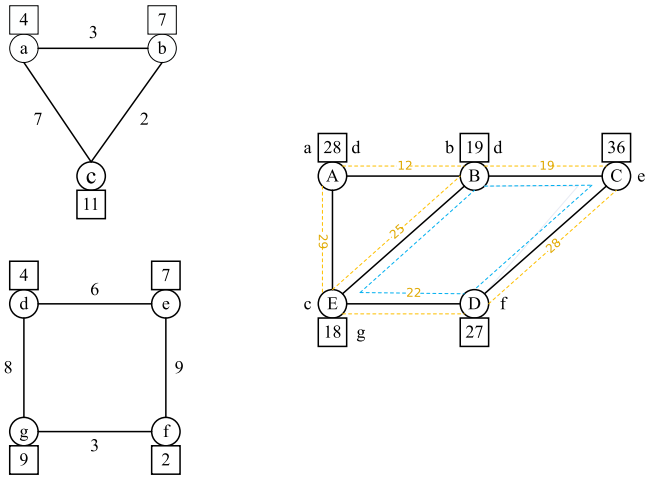


Fig. 1. An example of VNE.

B. Evaluation Metrics

To evaluate the utility of network embedding, we design three evaluation metrics in this paper. The utility of the VNE is related to the node resource expenditure $CPU(n^V)$ and the link resource expenditure $BW(l^V)$, which can be represented by:

$$R(G^V, t, t_d) = t_d \left[w_c \sum_{n^V \in N^V} CPU(n^V) + w_b \sum_{l^V \in L^V} BW(l^V) \right], \tag{5}$$

where w_b and w_c indicate the revenue weights of links and nodes, and t_d denotes the duration time of request G^V. Note that the value of w_c and w_b are only determined by the VNE. Correspondingly, the resource consumed by G^V can be represented by:

$$C(G^V, t, t_d) = t_d \left[\sum_{n^V \in N^V} CPU(n^V) + \sum_{l^V \in L^V} \sum_{l^S \in L^S} BW(f_{l^S}^{l^V}) \right], \tag{6}$$

where $BW(f_{l^S}^{l^V})$ indicates the bandwidth consumed by l^V in the underlying link l^S, and $CPU(n^V)$ represents the node resources expended by n^V.

Based on this, we design three evaluation metrics. Firstly, we design an average revenue metric to evaluate the profitability of the algorithm, which can be represented by:

$$Rev = \frac{\sum_{t=0}^{T} R(G^V, t, t_d)}{T}, \tag{7}$$

where T is the time interval.

Then, we design the revenue to cost metric, which can be formulated by:

$$Rev2Cos = \frac{\sum_{t=0}^{T} R(G^V, t, t_d)}{\sum_{t=0}^{T} C(G^V, t, t_d)}. \tag{8}$$

This metric indicates the resource utilization efficiency of algorithm. Meanwhile, the embedding success ratio of virtual

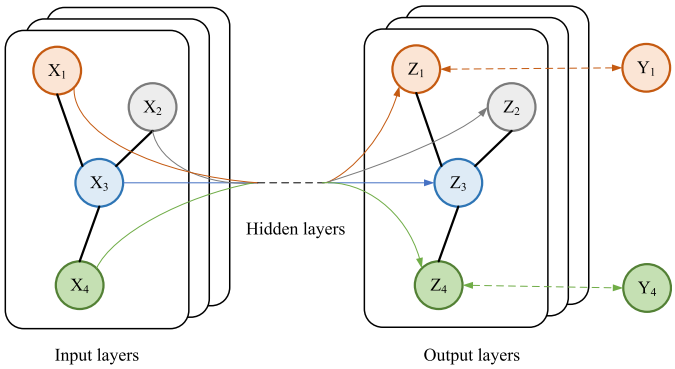


Fig. 2. The GCN architecture.

requests is another important indicator of the algorithm reliability. Thus, we design the average acceptance ratio metric, which can be represented by:

$$Acp = \frac{\sum_{t=0}^{T} Acp\ (G^V, t, t_d)}{\sum_{t=0}^{T} All\ (G^V, t, t_d)}, \tag{9}$$

where $Acp\ (G^V, t, t_d)$ is the number of successfully embedded VNs and $All\ (G^V, t, t_d)$ represents the total number of requests.

IV. GCN Aided Embedding Algorithm

In this section, we propose a GCN aided VNE algorithm. Also, the parameters of GCN convolution kernel are optimized by introducing policy gradient algorithm.

A. Graph Convolutional Network

GCNs are considered as a basic variant of graph neural networks. They are used to process non-Euclidean graph-structured data and is able to extract the high-order spatial features of topological graphs by passing a filter over the graph. The GCN is shown in Fig. 2, it adopts a multi-layered neural network to perform convolutional computation. In GCN, a graph $G = (V, E)$ is defined, where E and V denote the link and node sets of G. We use the Laplace matrix to perform the spectral decomposition to the graph, which is homologous with the spectral domain in GCN. The Laplace matrix is calculated by:

$$L = D - A, \tag{10}$$

where A denotes the adjacency matrix and D denotes the degree matrix and is the n-th eigenvalue. Also, the symmetric normalized Laplace can be formulated as:

$$L^{sys} = D^{-1/2} A D^{-1/2}. \tag{11}$$

Considering that the symmetric Laplace matrix satisfies the condition of spectral decomposition, the spectral decomposition form of the Laplace matrix can be represented by:

$$L = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^{-1}, \tag{12}$$

where $U = (\vec{u_1}, \vec{u_2}, \ldots, \vec{u_n})$ is a matrix with the unit eigenvector and λ_n is the n-th eigenvalue.

In addition, the Fourier transform is used for signal conversion between time domain and frequency domain. The Fourier transform formula can be given by:

$$F(w) = \mathcal{F}[f(t)] = \int f(t) e^{-iwt} dt, \tag{13}$$

where $f(t)$ is the signal in the time domain, and the e^{-iwt} satisfies:

$$\triangle e^{-iwt} = \frac{\partial^2}{\partial t^2} e^{-iwt} = -w^2 e^{-iwt}, \tag{14}$$

where the e^{-iwt} is the eigenfunction of the Laplace operator $\triangle$. According to the correlation, we can define the Fourier transform on the graph as:

$$F(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^{N} f(i) u_l^*(i), \tag{15}$$

where $u_l^*(i)$ is the $i\text{-}th$ component of the $l\text{-}th$ eigenvector.

Note that the above inner product operate in complex space, the $u_l^*(i)$ is the conjugate of $u_l(i)$. Thus, the Fourier transform on the graph is extended by matrix multiplication as:

$$\hat{f} = U^T f. \tag{16}$$

Also, the inverse Fourier transform of f on the graph can be formulated by:

$$f = U \hat{f}. \tag{17}$$

According to the convolution theorem, the convolution of $f(t)$ and $h(t)$ can be rewritten as:

$$(f * h)_G = U \begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix} U^T f. \tag{18}$$

Specifically, convolution parameter can be represented by $diag(\hat{h}(\lambda_l))$. And the $diag(\hat{h}(\lambda_l))$ is regarded as convolution kernel $diag(\theta_l)$ in:

$$y_{output} = \delta(U g_\theta(\Lambda) U^T x), \tag{19}$$

where $g_\theta(\Lambda) = \begin{pmatrix} \theta_1 & & \\ & \ddots & \\ & & \theta_n \end{pmatrix}$. The equation. 19 is the first generation of GCN propagation formula. Considering the computation and parameter complexity, we introduce Chebyshev polynomial to fit the GCN convolution kernel [30]. By using the Chebyshev polynomial, the following formula can be rewritten as:

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \beta_k T_k(\tilde{\Lambda}), \tag{20}$$

where $T_k(\tilde{\Lambda})$ denotes the order k of Chebyshev polynomials, β_k is the corresponding coefficient, and $\tilde{\Lambda}$ represents the diagonal matrix of eigenvalues after reprocessing and scaling. The reason for this re-scaled transformation is that the form of the first-generation chebyshev polynomial is $T_k(x) = cos(k \cdot arccos(x))$ and the input should be between (-1,1). Because the laplacian matrix is a positive semi-definite matrix with non-negative eigenvalues, after dividing by the maximum eigenvalue, and subtracting an identity matrix I, the goal is achieved. The transformation can be expressed by: $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I$. By substituting the equation. 20 into the equation. 19, the equation. 19 can be reformulated as:

$$y_{output} = \delta\left(U \sum_{k=0}^{K-1} \beta_k T_k(\tilde{\Lambda}) U^T x\right). \tag{21}$$

Then, because $L = U\Lambda U^T$, the equation. 21 can be rewritten as:

$$y_{output} = \delta\left(\sum_{k=0}^{K-1} \beta_k T_k(\tilde{L}) x\right), \tag{22}$$

where $\tilde{L} = 2\ L/\lambda_{max} - I$.

When $K = 2$, the propagation formula can be expressed as:

$$\begin{aligned} U g_\theta(\Lambda) U^T x \approx \sum_{k=0}^{1} \theta_k T_k(\tilde{\Lambda}) x &= \theta_0 x + \theta_1 (L - I) x \\ &= \theta_0 x - \theta_1 D^{-1/2} A D^{-1/2} x. \end{aligned} \tag{23}$$

In addition, an assumption $\theta = \theta_0 - \theta_1$ is introduced to our model. The equation. 23 can be adapted as:

$$U g_\theta(\Lambda) U^T x \approx \theta(I + D^{-1/2} A D^{-1/2}) x. \tag{24}$$

Moreover, to prevent the gradient dispersion caused by the repeated multiplication of $I + D^{-1/2} A D^{-1/2}$, we use the regularization trick:

$$I_N + D^{-1/2} A D^{-1/2} \rightarrow \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}, \tag{25}$$

where $\tilde{A} = A + I_N$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. The GCN propagation formula can be expressed as:

$$Y = \delta(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} X \Theta), \tag{26}$$

where $X \in R^{N \times C}$ represents the input node matrix, $\Theta \in R^{C \times F}$ is the parameter matrix of filters, and $Y \in R^{N \times F}$ represents the node matrix after convolution.

B. Information Extraction

In this paper, we selected four feature vectors to represent the physical node state as the first layer input.

1) Computational Capability (CPU) : $CPU(n^s)$ represents the remaining computational capability of node n^s, which can be calculated by:

$$CPU(n^s) = CPU(n^s) - \sum_{n^v \rightarrow n^s} CPU(n^v), \tag{27}$$

where $CPU(n^s)$ represents the remaining resource of n^s. The $\sum_{n^v \rightarrow n^s} CPU(n^v)$ denotes the sum of CPU requirements of n^v.

2) Bandwidth Sum (SUM) : The sum of bandwidth of all links connected to the node n^s can be calculated by:

$$SUM(n^s) = \sum_{l^s \in L^{n^s}} BW(l^s), \tag{28}$$

where L^{n^s} denotes all substrate links connected to n^s. Specifically, a higher $SUM(n^s)$ represents a higher probability of satisfying the link embedding.

3) Degree (DEG) : DEG denotes the number of links connected to this node. The DEG can be represented by:

$$DEG(n^s) = \sum_{n \in N^s} L(n^s, n), \tag{29}$$

where $L(n^s, n)$ means whether n^s and n are connected. The $L(n^s, n)$ is 1 if connected or 0 if not connected.

4) Distance from other embedded nodes (DST) : In the process of network embedding, we also need to consider the embedding location of other virtual nodes. The distance between nodes in the same request is closer, the fewer bandwidth resources will be consumed. In this paper, we introduce the FloydWarshall [32] algorithm to calculate the shortest distance between two nodes. The DST can be calculated by dividing the total distance by the number of nodes, which can be written as:

$$DST^{AVG}(n^s) = \frac{\sum_{\tilde{n^s} \in \tilde{N^s}} DST(n^s, \tilde{n^s})}{\left|\tilde{N^s}\right| + 1}, \tag{30}$$

where $\tilde{N^s}$ denotes the set of substrate nodes that the current virtual request has embedded, and $DST(n^s, \tilde{n^s})$ denotes the shortest distance between substrate nodes n^s and $\tilde{n^s}$.

Then, these four features can form the feature vector F of physical nodes:

$$F = [CPU, SUM, DEG, DST]. \tag{31}$$

C. GCN-VNE Algorithm

In this section, we proposed our GCN-VNE algorithm. In this paper, we adopt a typical two-stage embedding mechanism, which means nodes and links are mapped separately. The GCN propagation formula can be expressed by:

$$Y = \delta(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} X \Theta), \tag{32}$$

where $\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$ is a fixed value. For simplification, we consider $\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$ as A, and the formula can be rewritten as:

$$H^{l+1} = ReLU(A H^l W), \tag{33}$$

where H^l is the input matrix and W is the convolution kernel matrix of the l-th layer. In particular, H_0 is the initial feature matrix. Then, the first layer formula of GCN can be expressed as:

$$H^1 = ReLU(AH^0W^0)$$
$$(A \in R^{n\times n}, H^0 \in R^{n\times nfeat}, W^0 \in R^{nfeat\times nhidden}). \quad (34)$$

where $nfeat$ and $nhidden$ is the dimension of feature matrix and the first convolutional kernel, and the dimension of output H^1 can be calculated as $n \times nhidden$. Then, the output H^1 will be input to the last layer, which can be described as follows:

$$H^2 = ReLU(AH^1W^1)$$
$$(A \in R^{n\times n}, H^1 \in R^{n\times nhidden}, W^1 \in R^{nhidden\times nClass}), \quad (35)$$

where $nclass$ represents the number of categories. Finally, the dimension of H^2 can be calculated as $n \times nClass$, and the probability classification of each node can be obtained by the node classification task. But in VNE, our goal is to get the embedding result of the current virtual node. Therefore, the dimension of convolution kernel matrix W^1 needs to satisfy the condition $W^1 \in R^{nhidden\times 1}$ to meet the scenario requirements of VNE.

D. Policy Network

In the VNE scenario, our training data only contains network resources and structural information, which lack supervised labels. Therefore, we introduce the policy gradient algorithm to optimize the GCN model [33]. The policy π can be formulated as an equation with θ:

$$\pi_\theta(s,a) = P(a|s,\theta) \approx \pi(a|s). \quad (36)$$

The optimization objective function of the policy gradient is the expectation of the reward obtained. And a gradient ascent method is applied to optimize the objective function. The gradient of policy π_θ can be expressed as:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\ [\nabla_\theta log\pi_\theta(s,a)Q_\pi(s,a)], \quad (37)$$

where the $Q_\pi(s,a)$ denotes the the estimated reward value obtained by action a in state s. Then parameter updating formula of policy gradient can be given by:

$$\theta = \theta + \alpha\nabla_\theta \log \pi_\theta(s,a)Q_\pi(s,a). \quad (38)$$

where α denotes the learning rate of network parameters. Based on this, we present a policy network with GCN, which is shown in Fig. 3. The input feature matrix X of GCN can represent the state s, and the policy π_θ is constructed by the convolution kernel. Besides, we select the softmax function as the policy function, which can be described as:

$$\pi_\theta(s,a) = \frac{e^{\phi(s,a)^T\theta}}{\sum_b e^{\phi(s,b)^T\theta}}. \quad (39)$$

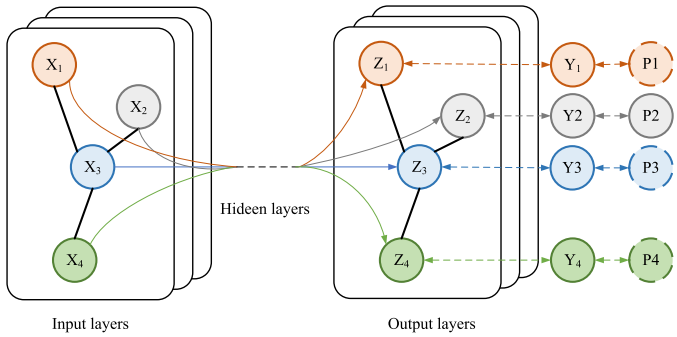


Fig. 3. The policy network.

E. Training and Testing

As shown in Fig. 3, our policy network consists of a GCN with two hidden layers. In each layer, the adjacent matrix A is static, and the input matrix X and convolution kernel matrix W are dynamic changed. We randomly select the label for each embedding result. And we manually define a label vector y with n dimensions for all virtual nodes, where n is equal to the nodes in the substrate network. Suppose that the node n_i is selected as the embedding label of the current virtual node, where only the i-th dimension of the label vector y is 1, and the other dimensions are 0. The output vector $\hat{y}$ of the policy network is a vector with n dimensions, which will be processed by softmax function:

$$p_i = \frac{e^{\hat{y}_i}}{\sum_j e^{\hat{y}_j}}, \quad (40)$$

where p_i constitutes the output vector p of the policy network. Then, the output vector p and label vector y can be applied to calculate a cross-entropy as the loss of our model:

$$L(y,p) = -\sum_i y_i log(p_i). \quad (41)$$

In the policy gradient algorithm, the reinforcement learning agents update policies based on reward signal [14]. The revenue to cost metric is set as the reward signal in this paper. In policy networks, the formula of parameter gradient update can be expressed as:

$$g = \alpha \cdot r \cdot g_f, \quad (42)$$

where r represents the reward signal, and g_f is the gradient calculated from the loss function. Note that if the link embedding fails, the gradient of loss g_f will not participate in the parameter update process, because the reward signal r can't be calculated.

Moreover, the learning rate α is crucial in determining the span of training. A small learning rate may cause slow convergence, while a big learning rate may cause the training unstable and over-fitting [34]. During the optimization process, we use the mini-batch SGD algorithm to train our policy network. It combines the advantages of batch gradient descent and random gradient descent algorithms to ensure the convergence speed and improve the training effect of the model.

The training process of GCN-VNE is shown in Algorithm 1. Note that lines 1-5 are the node embedding process, and lines 6-

Algorithm 1: Training of GCN-VNE.

Require: $epochNum$; $trainingSet$; $batchSize$;
Ensure: Trained GCN parameters;
1: GCN initialization;
2: **while** $epoch < epochNum$ **do**
3: num=0
4: **for** $request \in trainingSet$ **do**
5: **for** $node \in request$ **do**
6: $node_{host}$= GCN-VNE(node);
7: **end for**
8: **if** mapped ($\forall$ node $\in request$) **then**
9: **for** $link \in request$ **do**
10: $link_{host}$= Floyd(link);
11: **end for**
12: **end if**
13: **if** mapped ($\forall$ node $\in request$, $\forall$ link $\in request$) **then**
14: store g_f
15: **else**
16: delete g_f;
17: **end if**
18: num++
19: **if** num reaches the $batchSize$ **then**
20: Parameter update;
21: num=0;
22: **end if**
23: **end for**
24: $epoch$++;
25: **end while**

Algorithm 2: Testing of GCN-VNE.

Require: $testingSet$; Trained GCN parameters;
Ensure: Embedding result;
1: **for** $request \in testingSet$ **do**
2: **for** $node \in request$ **do**
3: $node_{host}$= GCN-VNE(node);
4: **end for**
5: **if** mapped ($\forall$ node $\in request$) **then**
6: **for** $link \in request$ **do**
7: $link_{host}$= Floyd(link);
8: **end for**
9: **end if**
10: **if** mapped ($\forall$ node $\in request$, $\forall$ link $\in request$) **then**
11: return result;
12: **else**
13: break;
14: **end if**
15: **end for**

8 are the link embedding process. For all embedded nodes, the Floyd algorithm is used to find the closest path among them. The algorithm calculates the gradient in lines 9-12. After the training, we obtained a trained policy network. Then we use the testing set to evaluate the embedding policy performance. Algorithm 2 is our testing algorithm pseudo-code of GCN-VNE. In the process of node embedding, the node with the maximum probability in the output vector of the policy network is applied as the embedding node. After all nodes in the request are successfully embedded, the link embedding starts. Similarly, the Floyd algorithm is used in the link embedding in the testing. Any failure in the process will cause the embedding to fail.

F. Computational Complexity Analysis

In this section, we analyze the computational complexity of our proposed GCN-VNE training algorithm. For simplification, we assume that the number of virtual requests in the training data is r, and the average number of virtual nodes and links in each request is m and n. The function f(n) represents the execution frequency of the training algorithm. During the node embedding phase,the convolution kernel of the GCN network is fitted by the second-order Chebyshev polynomial to perform the forward calculation, and the computational complexity is simplified to a constant term. After the node embedding phase, we apply the Floyd algorithm to compute the shortest path between nodes during the link embedding phase, and the computational complexity can be calculated as n^3. After that, the calculation and storage of gradient and the updating of network

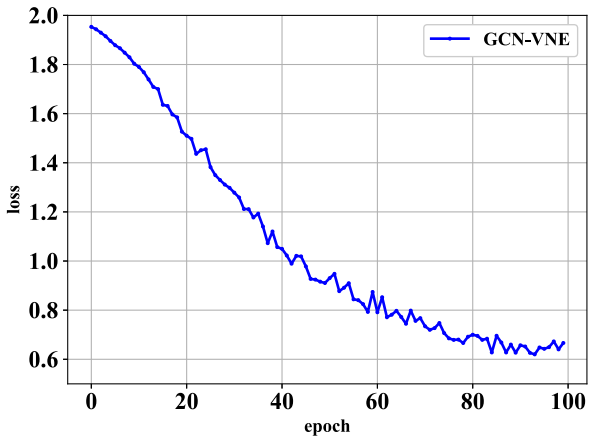


Fig. 4. Loss on the training set.

parameters belong to the computational complexity of constant terms. Therefore, the final function $f(n)$ can be expressed as:

$$f(n) = r(m + n^3) \tag{43}$$

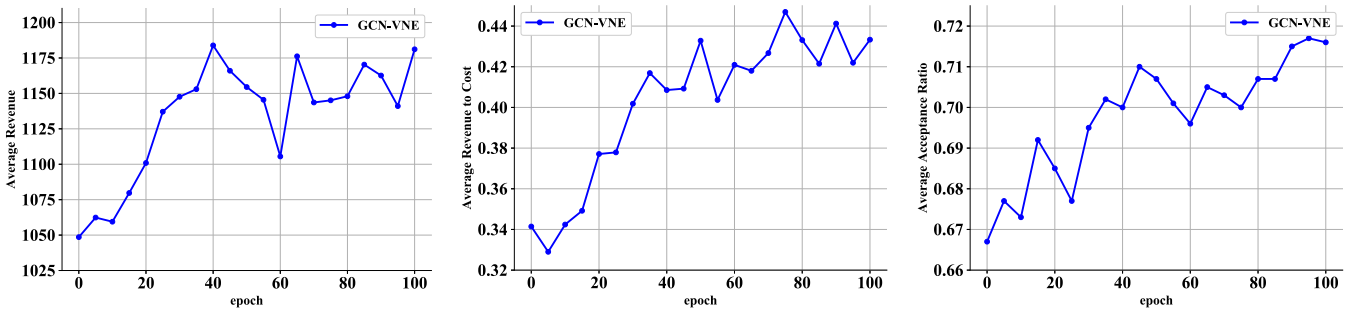
and the computational complexity after simplification is $O(rn^3)$.

V. Simulation Results

In this section, we give the experiment results of our GCN-VNE algorithm.

A. Experiment Setup

In this part, we present the details of the experiment setting. We use the GT-ITM to generate the network [35]. The experiment topology contains 550 links and 100 nodes. The capacity of computational resources in each node is following a uniform distribution in (50,100), and the bandwidth are according to a uniform distribution between (20,50). Besides, we generate two virtual network requests sets, each contains 1000 requests. One set is used in training and the other is used in testing. Specifically, the number of nodes in each VN is according to the



(a) Revenue in training (Eq. (7)). (b) Revenue to cost ratio in training (Eq. (8)). (c) Acceptance ratio in training (Eq. (9)).

Fig. 5. The algorithm performance on the training set.

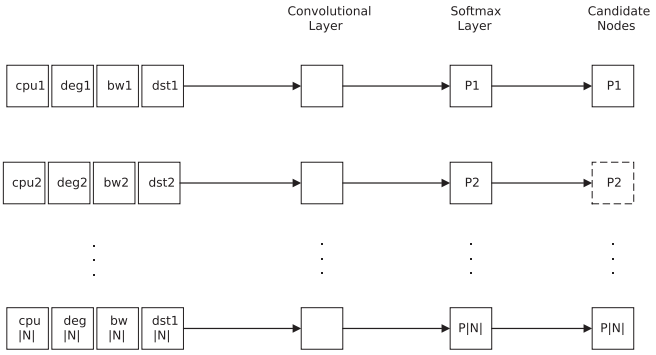


Fig. 6. The RLVNE model.

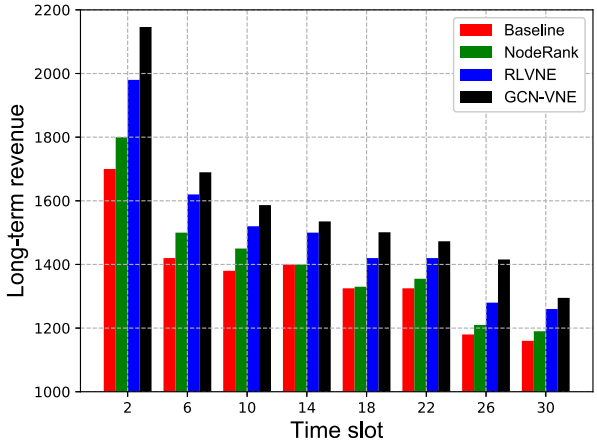


Fig. 7. Revenue in testing (Eq. (7)).

uniform distribution between (2,10), and the computing resources follow the uniform distribution between $(0, 50)$. The bandwidth resources of each virtual link are evenly distributed between (0,50). The connection probability between links is set as 0.5, indicating that the average number of connections per virtual request is $n(n-1)/4$. The arrival time of virtual requests follows the Poisson distribution. On average, 4 requests arrive within 100 time-units. The duration of each request is exponentially distributed, with an average duration of 1000 time units.

B. Convergence Analysis

We first evaluate the convergence of our proposed algorithms. As shown in Fig. 4, the loss is decreasing during the training time. At the beginning of training, we notice that the loss decreases rapidly. This is caused by the sufficient optimization space brought by the parameter randomization. As continuously training, the decline speed of loss tends to be gentle. In about 90 to 100 epochs, the loss is almost constant. The simulation results show that our GCN-VNE can effectively learn and converge to the optimal point.

Besides, we demonstrate the change of three evaluation metrics during the training. As shown in Fig. 5, we notice that three metrics are at all relatively low value at the beginning of training. This is because we adopt randomly initialized parameters in the policy network. Along with the training, the embedding mechanism is optimized gradually and the metrics are improving steadily. After about 80 epochs, due to the limitation of substrate resources, each metric reaches a stable optimal state.

C. Performance Analysis

It can be inferred from the metric graphs in the training that the training effect of the GCN-VNE is relatively believable, and the benefits of each metric are obvious. In the second experiment, we compare the performance of our algorithm with the baseline algorithm in the test set. We adopt three baseline VNE algorithms in this paper. The first baseline is the algorithm presented in [10], which introduced a series of path optimization policies to optimize the link embedding and significantly reduces the bandwidth consumption. The second baseline is the NodeRank algorithm in [12]. In NodeRank, the resource availability of nodes can be calculated by $H(n^s)$:

$$H(n^s) = CPU(n^s) \sum_{l^s \in L(n^s)} BW(l^s). \tag{44}$$

The last baseline algorithm is the RLVNE presented in [36], where a policy network based CNN is used to optimize the embedding decision. Fig. 6 is the policy network architecture of the RLVNE. A comparison with the RLVNE algorithm can intuitively show the difference between CNN and the GCN in extracting spatial features of topological graphs.

As shown in Figs. 7, 8, 9, we notice that the variation trend of the four algorithms is roughly consistent. Specifically, the acceptance ratio and revenue metrics are high at the beginning of the testing, which is due to the sufficient resources of the substrate network. With the continuous mapping of requests, these

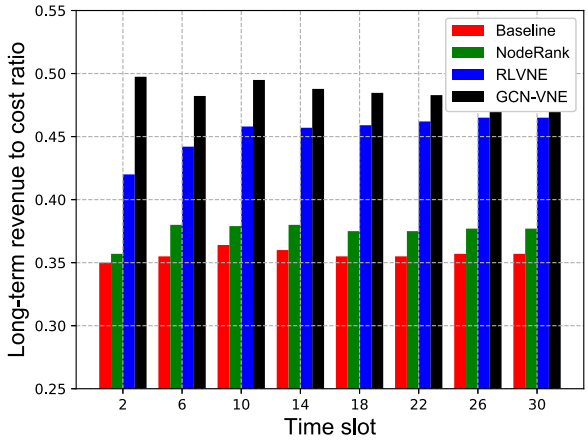


Fig. 8. Revenue to cost ratio in testing (Eq. (8)).

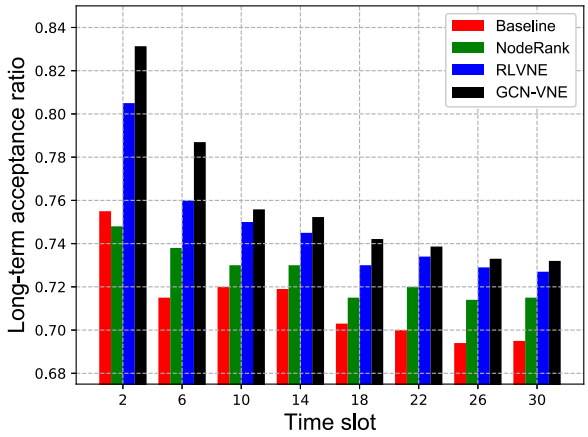


Fig. 9. Acceptance ratio in testing (Eq. (9)).

two metrics are declined accordingly. In contrast, the revenue to cost metric does not fluctuate during the testing because it has no relation with the amount of the substrate resources available [37]. Moreover, in the testing graphs, our algorithm is always above other algorithms. Therefore, we can conclude that our GCN-VNE is the best of these four algorithms.

VI. Conclusion

In this paper, we apply network virtualization to IoT, where the NV enables multiple virtual network applications to share the same physical network, thus providing IoT users with various customized end-to-end services. Based on this, we propose a GCN aided VNE algorithm to extract the high-order relationship between nodes in the substrate network. The GCN is modified according to the features of VNE. Besides, we introduce the policy gradient algorithm and build a policy network based on GCN to optimize the VNE performance. We also design three evaluation metrics to evaluate the utility of the network embedding policy. The experiment results show that our GCN-VNE algorithm outperforms some state-of-the-art schemes.

References

[1] S. Ran, "A model for web services discovery with QoS," *ACM Sigecom Exchanges*, vol. 4, no. 1, pp. 1–10, 2003.

[2] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, Jan.–Mar. 2016.

[3] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, Feb. 2013.

[4] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource trading in blockchain-based industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3602–3609, Jun. 2019.

[5] D. G. Andersen, "Theoretical approaches to node assignment," Dec. 2002.

[6] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE 25th Int. Conf. Comput. Commun.*, Barcelona, Spain, 2006, pp. 23–29.

[7] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to Pareto-optimal wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1472–1514, Jul.–Sep. 2020.

[8] H. Yao, T. Mai, C. Jiang, L. Kuang, and S. Guo, "Ai routers & network mind: A hybrid machine learning paradigm for packet routing," *IEEE Comput. Intell. Mag.*, vol. 14, no. 4, pp. 21–30, Nov. 2019.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[10] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Apr. 2008.

[11] A. Razzaq and M. S. Rathore, "An approach towards resource efficient virtual network embedding," in *Proc. Int. Conf. Evolving Internet*, Valencia, Spain, 2010, pp. 68–73.

[12] X. Cheng et al., "Virtual network embedding through topology-aware node ranking," *Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.

[13] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on the degree and clustering coefficient information," *IEEE Access*, vol. 4, pp. 8572–8580, 2016.

[14] C. Qiu, H. Yao, F. R. Yu, F. Xu, and C. Zhao, "Deep Q-learning aided networking, caching, and computing resources allocation in software-defined satellite-terrestrial networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5871–5883, Jun. 2019.

[15] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE 28th Int. Conf. Comput. Commun.*, 2009, pp. 783–791.

[16] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. 1st ACM SIGCOMM Workshop Virtualized Infrastructure System Architectures*, Barcelona, Spain, 2009, pp. 81–88.

[17] S. Shanbhag, A. R. Kandoor, W. Cong, R. Mettu, and T. Wolf, "VHub: Single-stage virtual network mapping through hub location," *Comput. Netw.*, vol. 77, pp. 169–180, Dec. 2015.

[18] X. Hesselbach, J. R. Amazonas, S. Villanueva, and J. F. Botero, "Coordinated node and link mapping VNE using a new paths algebra strategy," *J. Netw. Comput. Appl.*, vol. 69, pp. 14–26, Apr. 2016.

[19] R. Mijumbi, J. L. Gorricho, J. Serrat, M. Claeys, J. Famaey, and F. D. Turck, "Neural network-based autonomous allocation of resources in virtual networks," in *Proc. Eur. Conf. Netw. Commun.*, Bologna, Italy, 2014, pp. 1–6.

[20] S. Haeri and L. Trajkovic, "Virtual network embedding via Monte Carlo tree search," *IEEE Trans. Cybern.*, vol. 48, no. 2, pp. 510–521, Feb. 2018.

[21] Y. Kawamoto, H. Takagi, H. Nishiyama, and N. Kato, "Efficient resource allocation utilizing Q-learning in multiple UA communications," *IEEE Trans. Netw. Sci. Eng.*, vol. 6, no. 3, pp. 293–302, Jul.–Sep. 2019.

[22] F. Tang, Z. M. Fadlullah, B. Mao, and N. Kato, "An intelligent traffic load prediction-based adaptive channel assignment algorithm in SDN-IoT: A deep learning approach," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5141–5154, May 2018.

[23] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.

[24] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.

[25] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.

[26] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "Struc2vec: Learning node representations from structural identity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 385–394.

[27] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 1024–1034.
[28] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Representations*, 2014, pp. 1067–1077.
[29] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, pp. 129–150, Dec. 2009.
[30] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[31] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Li, and Y. Bengio, "Graph attention networks," *Proc. Int. Conf. Learn. Representations*, vol. 1050, pp. 20–22, 2017.
[32] S. Hougardy, "The Floyd-Warshall algorithm on graphs with negative cycles," *Inf. Process. Lett.*, vol. 110, no. 8–9, pp. 279–281, 2010.
[33] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 37–4, pp. 229–256, 1992.
[34] F. Li, H. Yao, J. Du, C. Jiang, and Y. Qian, "Stackelberg game-based computation offloading in social and cognitive industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5444–5455, Aug. 2020.
[35] E. Z. M. Thomas, "Generation and analysis of random graphs to model internetworks," *College Comput. Georgia Inst. Technol.*, vol. 63, no. 4, pp. 413–442, Jul. 1994.
[36] H. Yao, H. Liu, P. Zhang, S. Wu, C. Jiang, and S. Guo, "A learning-based approach to intra-domain QoS routing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6718–6730, Jun. 2020.
[37] H. Yao, S. Ma, J. Wang, P. Zhang, C. Jiang, and S. Guo, "A continuous-decision virtual network embedding scheme relying on reinforcement learning," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 2, pp. 864–875, Jun. 2020.

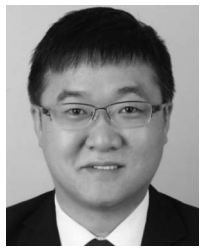
Sihan Ma received the bachelor's degree in 2018 from the Beijing University of Posts and Telecommunications, Beijing, China, where he is currently working toward the master's degree with the School of Information and Communication Engineering. His research interests include semantic computing and Big Data for networking.

Haipeng Yao (Senior Member, IEEE) received the Ph.D. degree with the Department of Telecommunication Engineering, University of Beijing University of Posts and Telecommunications, Beijing, China, in 2011. He is currently a Professor with the Beijing University of Posts and Telecommunications. He has authored or coauthored more than 100 papers in prestigious peer-reviewed journals and conferences. His research interests include future network architecture, network artificial intelligence, networking, space-terrestrial integrated network, network resource allocation, and dedicated networks. Dr. Yao was the Editor of IEEE NETWORK, IEEE ACCESS, and the Guest Editor of IEEE OPEN JOURNAL OF THE COMPUTER SOCIETY and *Springer Journal of Network and Systems Management*. He was a Member of the Technical Program Committee and the Symposium Chair for a number of international conferences, including IWCMC 2019 Symposium Chair, and ACM TUR-C SIGSAC2020 Publication Chair.

Tianle Mai (Student Member, IEEE) is currently working toward the Ph.D. degree with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China. His research interests include future network architecture, network artificial intelligence, multi-agent system, space-terrestrial integrated network, network resource allocation, and dedicated networks.

Jingkai Yang (Student Member, IEEE) received the bachelor's degree and is currently working toward the Ph.D. degree with the Beijing University of Posts and Telecommunications, Beijing, China. His research interests include future network, network resource allocation, and dedicated networks.

Wenji He (Student Member, IEEE) is currently working toward the graduation degree with the Beijing University of Posts and Telecommunications, Beijing, China. Her research interests include computer working and the security of the Internet.



Kaipeng Xue (Senior Member, IEEE) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003, and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. He is currently an Associate Professor with the Department of EEIS and the School of Cyber Security, USTC. He has authored and coauthored more than 80 technical articles in the areas of communication networks and network security. His research interests include next-generation Internet, distributed networks, and network security. He is an IET Fellow. His work was the recipient of best paper awards in the IEEE MSN 2017, IEEE HotICN 2019, and Best Paper Runner-Up Award in the IEEE MASS 2018. He serves on the Editorial Board of several journals, including the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, *Ad Hoc Networks*, IEEE ACCESS, and *China Communications*. He is the Program Co-Chair of the IEEE IWCMC 2020 and SIGSAC@TURC 2020.

Mohsen Guizani (Fellow, IEEE) received the B.S. (Hons.) and M.S. degrees in electrical engineering and the second M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a Professor and the associate provost with the Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. He is the author of nine books and more than 600 publications in refereed journals and conferences. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid. He is a Senior Member of ACM. Throughout his career, he was the recipient of three teaching awards and four research awards. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He was the IEEE Computer Society Distinguished Speaker. He is currently the IEEE ComSoc Distinguished Lecturer. He guest edited a number of special issues in IEEE journals and magazines. He is also the Editor-in-Chief of the *IEEE Network Magazine*. He serves on the editorial boards for several international technical journals and the Founder and the Editor-in-Chief for *Wireless Communications and Mobile Computing journal* (Wiley).