IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 15 September 2023, accepted 9 October 2023, date of publication 12 October 2023, date of current version 20 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3324222

RESEARCH ARTICLE

An Attention Mechanism-Based Microservice Placement Scheme for On-Star Edge Computing Nodes

XIANGYU SU[1], AMR TOLBA[2], (Senior Member, IEEE), YUXI LU[3], LIZHUANG TAN[4], JIAN WANG[5], (Senior Member, IEEE), AND PEIYING ZHANG[3,4], (Member, IEEE)

[1]School of Information and Communications, Shenzhen Institute of Technology, Shenzhen 518116, China
[2]Department of Computer Science, Community College, King Saud University, Riyadh 11437, Saudi Arabia
[3]College of Computer Science and Technology, Qingdao Institute of Software, China University of Petroleum (East China), Qingdao 266580, China
[4]Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250013, China
[5]College of Science, China University of Petroleum (East China), Qingdao 266580, China

Corresponding authors: Amr Tolba (atolba@ksu.edu.sa) and Lizhuang Tan (tanlzh@sdas.org)

This work was supported in part by the Researchers Supporting Project, King Saud University, Riyadh, Saudi Arabia, under Grant RSPD2023R681; in part by the Natural Science Foundation of Shandong Province under Grant ZR2023LZH017, Grant ZR2022LZH015, and Grant 2023QF025; in part by the Industry-University Research Innovation Foundation of Ministry of Education of China under Grant 2021FNA01001; in part by the Integrated Innovation of Science, Education and Industry of Qilu University of Technology (Shandong Academy of Sciences) under Grant 2023PX057; and in part by the Talent Project of Qilu University of Technology (Shandong Academy of Sciences) under Grant 2023RCKY141.

ABSTRACT In the context of high-speed networks with 5G and 6G, the influx of user requests under variable usage scenarios puts great pressure on the monolithic architecture, and quality of service (QoS) is gradually not guaranteed. Placing low-coupling, high-efficiency microservices on satellite edge computing nodes with wide coverage is a good solution, but the exponential increase of users and edge nodes accessing communication networks in recent years has gradually highlighted the importance of proper placement and effective management of microservices. The existing studies generally fail to achieve autonomous management of microservices in a variable and complex network environment, and the few studies on autonomous management of microservices are limited to achieving autonomous placement without constraints among microservices. The quality of service and operation cost will not be guaranteed when facing a large number of network requests at the same time. This paper addresses the much-needed problem of modeling microservice placement in satellite edge nodes as a network embedding problem and effectively captures the features that affect microservice placement performance using the attention mechanism in graph neural networks. Simulation experimental results illustrate the effectiveness of the research content of this paper for the automatic management of microservices in satellite networks, while the proposed scheme in this paper performs well in terms of success rate and the benefit-overhead ratio of microservice placement.

INDEX TERMS Software defined networking (SDN), edge computing, microservice management.

I. INTRODUCTION

In recent years, the space-air-ground integrated network has garnered significant attention within the communication field as an emerging network architecture [1], [2]. Within this novel network framework, two key technologies have emerged as pivotal for future development: Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technology. SDN, rooted in cloud technology [3], embraces the concept of segregating control and forwarding functionalities [4]. It conceptualizes the entire network as a vast resource reservoir [5], enabling the efficient

The associate editor coordinating the review of this manuscript and approving it for publication was Alba Amato.

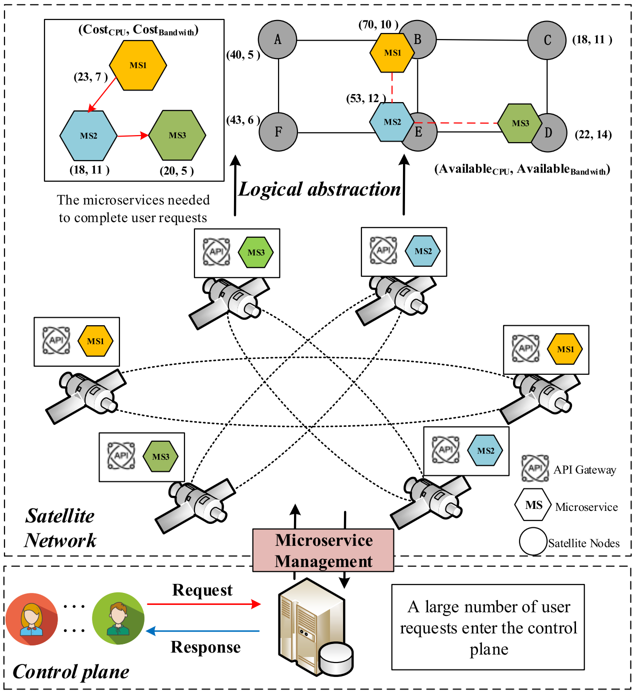


FIGURE 1. The process of autonomous management of microservices in satellite networks.

scheduling and allocation of resources. It also facilitates the reconfiguration of network functions and swift deployment. Microservices are a new technology that evolved from a single application architecture in recent years [6], [7]. Microservices vertically split the traditional monolithic system into multiple small functional components according to business requirements, and each microservice can be run and deployed separately [8]. The concept of microservice is analogous to the service function chain (SFC) in SDN, which is a logical link composed of virtual network function nodes in the order of functional requirements according to user needs [9]. The satellite network in the integrated air-space network has the characteristics of wide coverage and is not affected by natural disasters compared with the terrestrial network [10], [11], and deploying some microservices to the satellite network can effectively improve the quality of service (QoS) and the efficiency of edge computing [12], [13]. For instance, Dowhuszko et al. [14] proposed that telecom operators can store frequently requested content in 5G satellite edge nodes in advance during off-peak periods, which can reduce the placement time and effectively utilize network resources.

As satellite networks expand and the count of microservices increases, the potential combinations of microservice placements and assignments experience exponential growth [15]. This exponential surge leads to an extensive search space for exploring all feasible solutions, posing challenges for conducting a comprehensive search within a reasonable timeframe. Moreover, during the placement process of microservices, intricate considerations arise, including reciprocal dependencies among microservices, alongside numerous other factors such as the CPU and bandwidth resources available on satellite nodes. These considerations further compound the complexity of arriving at a solution. The virtual network embedding problem similar to the microservice placement problem has been shown to be an NP-Hard problem [16]. How to place each class of microservice instances on the appropriate satellite edge computing nodes in a controlled time while minimizing the capital expenditure (CAPEX) and operational cost (OPEX) of operators and guaranteeing the QoS of users is a hot topic of research [17], [18]. There are various ways to place and manage microservices, but when a large number of satellite nodes meet the placement conditions and a large number of user requests appear at the same time, it is difficult to manage the placement manually. Mayer et al. [19], [20], [21] tried to do research related to microservice management and monitoring, but unfortunately, they did not achieve fully autonomous management. A few studies have focused on the problem of autonomous placement of microservices, but they only consider the placement of individual microservice nodes and do not consider the dependencies before and after the microservices.

Microservice management consists of a set of operations and tasks that are used to manage and maintain various aspects of the microservice architecture. One of the key components is the controller, which is used to implement specific management functions for microservices. What we need to do is to allow the controller to control the placement of each microservice in the appropriate satellite node. The physical hierarchy of microservices is mainly divided from top to bottom into user layer, application layer, data storage layer, and infrastructure layer. Microservice management is a broader concept. It first includes the function of service registration and discovery. Microservices register their information with the management platform at startup, including service name, IP address, port number, and so on. Other microservices can query and discover available services through the management platform, thus enabling communication between services. Monitoring and logging also belong to the scope of microservice management, which is used to monitor the operation status, performance indicators, and abnormalities of microservices in real-time. In addition, microservice management also includes security management, deployment, and upgrading. In this paper, we focus on the placement of microservices, which is the interaction between microservices and the infrastructure layer. Deployment of microservices is a crucial part of microservice management, which is related to the stability of the cloud platform and the revenue of the operator. Microservice placement requests in real environments arrive continuously. Deployment of microservices depends on constraints such as containerized environment, orchestration tools, and security levels. There are differences in the environments in each satellite node and the types of microservices that can be hosted are different, but a satellite node can host several different types of microservices. Such a status quo also results in the diversity of placement schemes. Figure 1 shows an example of the placement of a set of microservices with forward and backward call relationships. To ensure service integrity, the same microservice can only be hosted and run in one satellite node. There are several constraints such as cost overheads that also need to be considered when microservices are placed. Our proposed microservice autonomy management platform models satellite nodes and microservices as a graph structure consisting of nodes and edges when there is a service request from a user or a change in the satellite node. The graph neural network is a hot topic in the field of data science and machine learning in recent years. Petar et al. [22] 2017 proposed the graph attention mechanism, which has the ability to learn from graph data and provide more accurate results. Problems such as resource scheduling [23], knowledge graphs, etc. can find solutions by combining with theories related to graph attention. Inspired by the recent research related to virtual network embedding in SDN, we propose an autonomous microservice placement strategy for on-satellite edge computing under software-defined networks. We mainly use the graph attention mechanism to aggregate the features between satellite nodes and microservice nodes to calculate the relative importance thus achieving better performance in the microservice placement problem.

Especially, the main contributions of this paper are as follows.

1) To address the problem of increased difficulty in operation and maintenance in satellite networks due to the possible emergence of a larger volume of microservices in the future, this paper designs a set of edge computing microservice management platform that operates autonomously in satellite networks according to the characteristics of satellite networks.
2) We model the microservice management problem as a candidate physical node selection problem. We define satellite nodes as physical network nodes, and the whole satellite network topology is regarded as a physical network, and use a deep learning approach to solve the microservice placement problem.
3) The simulation experiments demonstrate that the microservice placement strategy proposed in this paper achieves satisfactory results in terms of acceptance rate, average benefit ratio and comprehensive evaluation metrics.

The rest of this paper is summarized below. In Section II the related work is analyzed. Section III analyzes the problem in detail and proceeds to model the problem as. Section IV demonstrates the effectiveness of the work in this paper through simulation experiments. Section V summarizes the work done in this paper and points out the future research directions.

II. RELATED WORK

In this section, we introduce the relevant research progress in recent years from two aspects, namely, microservice management and applications of graph neural networks, respectively.

A. MICROSERVICE MANAGEMENT

Microservices are easy to develop and maintain and can be extended at a fine-grained level according to demand, which has become a hot research direction in recent years. Some of these researchers focus on the study of microservice monitoring. Mayer et al. [19] propose an experimental dashboard for microservice monitoring and management, which enables monitoring and management of microservice status, but the study cannot achieve autonomous placement and orchestration of microservices. Another part of the researchers optimizes for the subsequent stages after microservice placement. Jiang et al. [24] also proposed an idea for efficient management of microservice architectures, using technologies such as Redis clusters and service gateways to achieve load balancing of microservices, but still could not achieve fully autonomous management of microservices. Xu et al. [25] proposed an enhanced service framework based on microservice management for efficient access to services in edge computing environments. Li et al. [26] designed a fuzzy-based microservice computing resource scaling algorithm for a microservice management platform that can reduce the response time of microservice resource adjustment and achieve dynamic scaling of microservices both horizontally and vertically. Research on microservice management, as represented by Xu et al., has driven the development of edge computing, compared to researchers who have done relatively little research on the placement phase of microservices.

Earlier, an integer linear programming approach was proposed by formalizing the placement process as in Luizelli et al. [27] Zhang et al. [28] devised a solution method using a clustering algorithm combined with nonlinear programming. The joint edge server deployment and service placement model of this method formulates multiple constraints, such as the relationship between edge servers and base stations, storage capacity, and computational power of each edge server, to maximize placement profit. Tomassilli et al. [29] set minimizing the total deployment cost as the optimization objective. Zhao et al. address the current situation where existing studies do not consider service compliance attributes and propose a distributed redundant placement framework. This approach models the problem as a discrete stochastic optimization problem. Simulation results show that the scheme proposed by Zhao et al. [30] is robust. Overall, most of the existing algorithms for microservice placement focus on using linear integer programming and heuristic algorithms. These methods generally lack scalability and tend to fall into the category of optimal solutions.

Chen et al. [31] proposed a software-defined network-based virtual network embedding algorithm that first ranks physical nodes according to their importance and then maps virtual links between virtual nodes using a BFS policy. Chowdhury et al. [32] proposed to use of linear programming to map nodes while ensuring that the cost of embedding requests is as small as possible. However, the quality of the current embedding algorithms based on software-defined networks, when the current research handles embedding requests with a large influx of embedding requests, can be severely degraded.

B. APPLICATION OF GRAPH NEURAL NETWORKS

Graph Convolutional Neural Network (GCN) is a feature extractor similar to Convolutional Neural Network (CNN) [33], [34], but GCN is mainly oriented to graph-structured data.GCN is widely used in various directions related to graph data such as graph node classification, graph edge prediction, and graph embedding representation. The attention mechanism focuses on useful features in graph data, which can suppress useless information and can achieve efficient feature extraction, and reduce the difficulty of network training.

We proposed a scheme that can automate the control of microservices running in the satellite network under the software-defined network, inspired by the above-mentioned research, which can realize services such as placement, control, and maintenance of microservices.

III. PROBLEM DEFINITION AND ALGORITHM DESIGN

In this section, we model the problem of microservice placement for edge computing satellite nodes and describe in detail the algorithmic flow of microservice placement.

A. PROBLEM DEFINITION

In this paper, we list the relevant notations used in the problem definition in Table 1. Specifically, we abstract the satellite network as the physical network $SN = \{N^S, E^S, AV^S_{CPU}, AV^S_{Mem}, AV^S_{LBW}, AV^S_{LR}\}$. The properties of the satellite network include the available CPU of the satellite nodes, the memory, and the available bandwidth and resources of the links between the satellite nodes. Similarly, we define microservices as $MN = \{N^M, E^M, C^M_{CPU}, C^M_{Mem}, C^M_{LBW}, C^M_R\}$. Each microservice running in a satellite node consumes the CPU, memory, and bandwidth of the satellite node. Therefore, we define the microservice placement problem as $MN_i \to SN$. MN_i is a microservice placement request. The complete microservice placement process mainly includes the mapping of nodes and the mapping of dependencies between nodes.

We use the symbol $\varrho^{n^s}_{n^m_k}$ to indicate whether microservice n^m_k has been deployed on satellite node n^s, where k represents the k-th microservice in this group of microservices. Similarly, we use the symbol $\varsigma^{e^s}_{e^m_{pq}}$ to indicate whether the microservice p and q invocation process uses the communication link between satellite nodes. $\varrho^{n^s}_{n^m_k}$ and $\varsigma^{e^s}_{e^m_{pq}}$ are denoted as (1) and (2), respectively.

TABLE 1. Symbols related to the placement of microservices in the satellite network.

Symbol		Definition
Satellite Network	SN	Satellite network
	N^S	Nodes of satellite
	E^S	Links between satellite nodes
	AV^S_{CPU}	The available CPU capacity in satellite
	AV^S_{Mem}	The available memory in satellite
	AV^S_{LBW}	The available bandwidth of link between satellites
	AV^S_{LR}	The available resources of link between satellites
Microservices Network	MN	Microservices network
	N^M	Nodes of microservices
	E^M	Links between microservice nodes
	C^M_{CPU}	The required CPU to run microservices
	C^M_{Mem}	The required memory to run microservices
	C^M_{LBW}	Bandwidth occupied between microservice nodes
	C^M_R	Resources occupied between microservice nodes

$$\varrho^{n^s}_{n^m_k} = \begin{cases} 1, & \text{if } n^m_k \to n^s \\ 0, & \text{not deployed} \end{cases}$$
$$k = 1, 2, \ldots, Len(N^m),$$
$$n^m_k \in N^m, n^s \in N^s,$$
$$N^m \in MN_i, N^s \in SN. \tag{1}$$

$$\varsigma^{e^s}_{e^m_{pq}} = \begin{cases} 1, & \text{if } e^m_{pq} \to e^s \\ 0, & \text{not deployed} \end{cases}$$
$$e^m_{pq} \in E^m, e_s \in E^s,$$
$$E^m \in MN_i, E^s \in SN. \tag{2}$$

The following constraints also need to be satisfied when placing microservices:

$$AV^S_{CPU}\left(n^s\right) \geq C^M_{CPU}\left(n^m_i\right), if \quad \varrho^{n^s}_{n^m_i} = 1 \tag{3}$$
$$AV^S_{Mem}\left(n^s\right) \geq C^M_{Mem}\left(n^m_i\right), if \quad \varrho^{n^s}_{n^m_i} = 1 \tag{4}$$
$$AV^S_{LBW}\left(e^s\right) \geq C^M_{LBW}\left(e^m_{pq}\right), if \quad \varsigma^{e^s}_{e^m_{pq}} = 1 \tag{5}$$
$$AV^S_R\left(e^s\right) \geq C^M_R\left(e^m_{pq}\right), if \quad \varsigma^{e^s}_{e^m_{pq}} = 1 \tag{6}$$

(3)-(6) indicates that the total amount of resources remaining in the satellite nodes must be greater than or equal to the total amount of resources required by the microservices to be placed, and these constraints are satisfied to ensure the proper operation of the microservices,

$$\sum_{i=1}^{Len(N^S)} \varrho^{n^s_i}_{n^m_k} = 1 \quad \forall n^m_k \in N^M, \forall n^s_i \in N^S, \tag{7}$$

We use (7) to represent the uniqueness of microservice hosting. To ensure service integrity, one of the microservices in a set of microservices can only be hosted in one satellite node running at runtime. Where i denotes the i-th satellite node in the satellite network.

B. ALGORITHM DESIGN

Figure 2 shows the exact flow of our proposed microservice placement approach. The process mainly consists of feature extraction, calculating the importance using the trained model, sorting the nodes with the highest importance according to the calculated importance, and finally mapping the edges using the SPFA algorithm. Before starting the process, first the network properties of the satellite network and the microservice should be extracted separately. This is to allow the subsequent steps to be trained in a real network environment. The more attributes we extract, the more the computational complexity increases and the more redundant attributes may be extracted that do not have an impact on the placement. So instead of extracting all the attributes of the satellite network and microservices we extract the key attributes that affect the placement of the microservices. The key attributes include computational resources, storage resources, and bandwidth resources. Firstly, we need to construct feature vectors for the extracted network attributes using (8). Subsequently, we need to use these feature vectors to compute the attention weights between nodes.

$$\mathbf{h} = \left\{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\right\}, \vec{h}_i \in \mathbb{R}^F, \tag{8}$$

where the notation $\mathbb{R}^F$ is used to denote an F-dimensional real vector space, where each element is a real number, where F represents the dimensionality of the vector. the size of F is determined by the number of kinds of key attributes we extract from the network. $\mathbf{h}$ is a collection of feature vectors that contain the feature vectors of N nodes, and the feature vector $\vec{h}_i$ of each node belongs to an F-dimensional real feature vector space $\mathbb{R}^F$.

$$e_{ij} = a\left(\left[Wh_i \| Wh_j\right]\right), \tag{9}$$

The computation of Wh_i and Wh_j is a linear transformation, where Wh_i and Wh_j are the product between the features of node i and node j, respectively, and the matrix of learnable parameters W. This operation maps the original node features to a new feature space for subsequent attention computation. To increase the expressive power of the model, we introduce non-linear properties by introducing *LeakyReLu* to allow the model to learn more complex representations. In order to obtain the corresponding input and output transformations, we need to perform at least one linear transformation based on the input satellite node and microservice node features to obtain the output features, so we need to train a weight matrix $\mathbf{W} \in \mathbb{R}^{F' \times F}$ for all nodes. The learned weights W are carried out through the training data so that the model can be able to learn an effective representation of the graph data consisting of satellite nodes and microservices can be carried out. The attention factor formula is shown in (9), this formula can be expressed without considering the overall structure of the satellite network and microservice requests the importance of node j for node i. The symbol $\|$ has stitched for the node transformed features and finally maps the stitched high-dimensional features to a real number. The attention coefficient α_{ij} is obtained after normalising the real numbers using equation (10).

$$\alpha_{ij} = \text{softmax}_j\left(e_{ij}\right) = \frac{\exp\left(e_{ij}\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(e_{ik}\right)} \tag{10}$$

Next, we use Equation (11) to perform a weighted summation based on the computed attention coefficients.

$$h_i' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W h_j\right) \tag{11}$$

where $\mathcal{N}_i$ represents the combination of neighbor nodes of node i, in other words, the set of satellite nodes where this microservice can be placed. The weighted sum is computed based on the attention weight α_{ij} of the neighboring nodes to node i and makes it pass through the activation function of the obtained h_i' is the vector after aggregating the neighboring nodes. The training process pseudo code is shown in Algorithm 1. Lines 1-4 represent the initialisation process for the variables and the loop while wraps the training process. We set the number of training rounds to 100, the learning rate to 0.005, and the slope of LeakyReLu to 0.2. Figure 3 shows the Loss variation curve during the training process. It can be observed from the figure that the model experienced a rapid decline and eventually leveled off.

We propose an on-satellite microservice placement strategy in two main phases. In the first stage, the importance of each satellite node and microservice node at the current time is calculated and the microservice nodes in the service request are sorted in descending order according to their attention importance. Similarly, we have to sort all satellite nodes in order of importance. To ensure QoS, we finally use a greedy strategy to place the important microservice nodes on the satellite nodes with high importance values. In the second stage, the satellite nodes that cannot satisfy the bandwidth demand of the current service request are first offloaded. In the remaining satellite nodes, the mapping is completed using the SPFA algorithm. The detailed placement algorithm is shown in Algorithm 2 and Algorithm 3.

We assume that the number of microservice nodes is M and the number of satellite nodes is N. In the training phase, the time complexity is jointly determined by M and N as $O(MN)$. In the placement phase, the time complexity is jointly determined by the two phases of microservice node placement and link placement. In the microservice node placement phase time complexity is $O(MN)$. For the edge placement phase, the total computational complexity is $O(ED)$ assuming that the total number of edges is E and the number of edges on the path from a node that can reach the final destination node through a series of edges is D. The total computational complexity is $O(ED)$. So the total microservice placement complexity is $O(MN + ED)$.

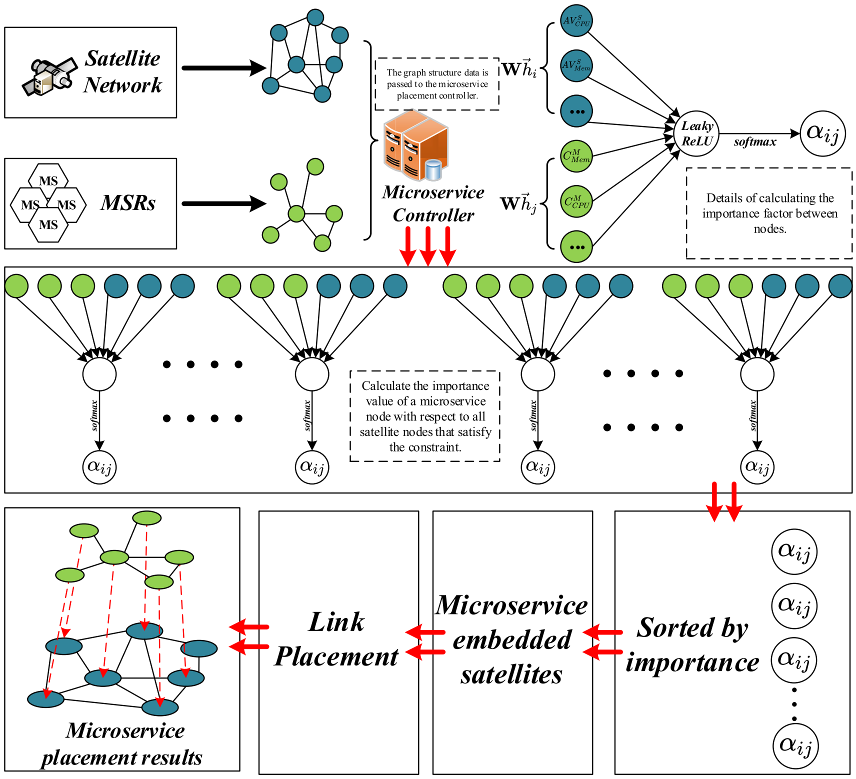


FIGURE 2. The process of placing microservices in satellite nodes using the attention mechanism.

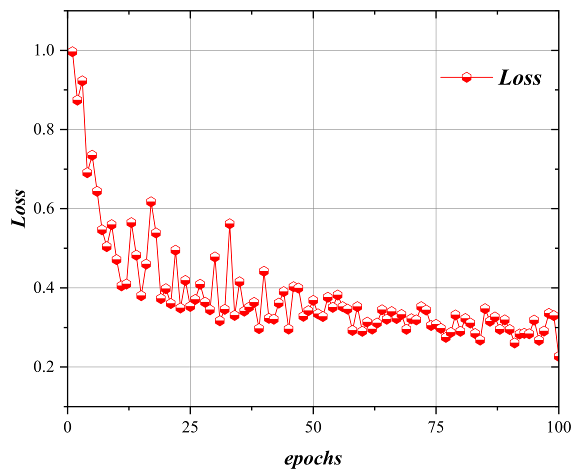


FIGURE 3. Training process loss curves.

C. EVALUATION INDICATORS

The main goal is to improve the utilization of satellite nodes as much as possible while ensuring the normal operation of microservices so that operators can gain more revenue. Since this problem is difficult to find the optimal solution in polynomial time, we usually design some evaluation metrics for the algorithm to evaluate its performance of the algorithm. In this paper, we use three metrics, long-term average revenue, acceptance rate, and comprehensive evaluation metrics, to evaluate the microservice placement strategy. We define the evaluation metrics as follows.

Algorithm 1 Training Process

Input: N^S, AV^S_{CPU}, AV^S_{Mem}, N^M, C^M_{CPU}, C^M_{Mem}, *epochs*
Output: Weighting matrix W
Initialize input feature dimension *in_features*;
Initialize output feature dimension *out_features*;
Initialize the *LeakyReLU* activated parameters *alpha*, dropout parameters *dropout*, *iteration*;
Initialize W according to AV^S_{CPU}, AV^S_{Mem}, N^M, C^M_{CPU} and C^M_{Mem};
while *iteration* $\leq$ *epochs* **do**
  foreach *Microservice nodes in microservice requests* **do**
    foreach *Each satellite node that can be placed* **do**
      Clear the gradient;
      Calculate Wh_i and Wh_j;
      Splice Wh_i and Wh_j and activate them using *LeakyReLU* using Equation (9);
      Use Equations (10) and (11) to compute the attention factor and aggregate neighbouring nodes;
      Calculated loss and back propagation;
      Update W;
  iteration++;

1) LONG-TERM REVENUE-COST RATIO

As shown in (12), we define the bandwidth requirement size of the link for each service request sent by each user, and the

Algorithm 2 Microservice Node Placement Algorithm

Input: N^S, AV^S_{CPU}, AV^S_{Mem}, N^M, C^M_{CPU}, C^M_{Mem}, W
The importance values between nodes is calculated by substituting the known weight matrix W and node attributes into Equations (8)-(10);
Rank the importance values of nodes N^S and N^M in descending order;
foreach $N^M = \{n^m_1, n^m_2, \ldots\}$ **do**
  foreach $N^S = \{n^s_1, n^s_2, \ldots\}$ **do**
    if *The node does not place the requested service and satisfies constraints (3) and (4)* **then**
      Placement of microservice nodes $n^m_i \rightarrow n^s_j$;
      Update satellite node n^s_j resource status;
      break;

Algorithm 3 Microservice Link Selection Algorithm

Input: N^S, E^M, AV^S_{LBW}, AV^S_{LR}, C^M_{LBW}, C^M_R
Resource requirements for the links between microservices requests C^M_{LBW} and C^M_R in descending order;
foreach *Unplaced links in* $E^M = \{e^m_{ij}, \ldots\}$ **do**
  foreach $E^S = \{e^s_{ij}, e^s_{pq}, \ldots\}$ **do**
    if *The resources remaining on this link do not satisfy constraints (5) and (6).* **then**
      Offload links in E^S;
    Put the first microservice node n^S_{first} to be placed in *MapQueue*;
    Place the first microservice node to be placed;
    while *MapQueue* $\neq \varnothing$ **do**
      Remove the set N^S_{next} of nodes connected to the satellite node n^S_{first} from *MapQueue*;
      foreach N^S_{next} **do**
        if *Joining this link will consume less resources than before* **then**
          Adding nodes to *MapQueue*;
    Update link resource status;

sum of the resource requirement size and the size of the CPU and memory computing resources required from the satellite node constitutes the operational gain for the satellite network from that request. We introduce the concept of *hops*, where additional spend exists if the placement process is performed across satellite nodes during placement. If this is not the case, the spend and gain are equal. (13) is the cost required to place the microservice. (14) is the long-term average benefit ratio equation.

$$\mathrm{REVE} = \sum_{n^m \in N^M} \left\{ C^M_{CPU}\left(n^m_i\right) + C^M_{Mem}\left(n^m_i\right) \right\} + \sum_{e^m \in E^M} C^M_{LBW}\left(e^m_{ij}\right) + \sum_{e^m \in E^M} C^M_R\left(e^m_{ij}\right) \tag{12}$$

$$\mathrm{COST} = \sum_{n^m \in N^M} \left\{ C^M_{CPU}\left(n^m_i\right) + C^M_{Mem}\left(n^m_i\right) \right\} + \Big(\sum_{e^m \in E^M} C^M_{LBW}\left(e^m_{ij}\right) + \sum_{e^m \in E^M} C^M_R\left(e^m_{ij}\right)\Big) * hops(e^m_{ij}) \tag{13}$$

$$RC = \lim_{t \to \infty} \frac{\sum_{t=0}^{T} REVE}{\sum_{t=0}^{T} COST} \tag{14}$$

2) ACCEPTANCE RATIO

The number of microservice chains successfully placed on satellite nodes based on user requests as a percentage of used user requests. A higher acceptance rate means that more user requests can be processed in the same situation, so a higher acceptance ratio is better.

$$\mathrm{ACC} = \lim_{t \to \infty} \frac{\sum_{t=0}^{T} Req_{acc}}{\sum_{t=0}^{T} Req_{all}} \tag{15}$$

3) COMPREHENSIVE EVALUATION INDICATORS

We define the weighted sum of the long-term average revenue and acceptance rate as the composite evaluation metric for microservice placement. In this paper, α takes the value of 0.3 and β takes the value of 0.7.

$$\mathrm{CE} = \alpha ACCR + \beta AVG_{reve} \tag{16}$$

IV. EXPERIMENT

In this section, we introduce the simulated simulation experimental environment and experimental conditions set up according to the real satellite network environment, and detail the evaluation metrics of this paper, and finally analyze the effectiveness of our proposed microservice placement strategy in the satellite network based on the experimental results.

A. EXPERIMENTAL ENVIRONMENT

The experiments in this paper were run on a host with an 8-core, 16-thread i7-11800H processor and 16GB-DDR4 3200MHz memory. The graphics card used for training in this article is an NVIDIA GeForce RTX3060 with 6G of video memory. In this paper, we used the ALEVIN2 tool to randomly generate a satellite network containing 150 nodes and 500 microservice placement requests. We exported the generated topological network as an XML format file and put it into the project for processing. The programming environment we used is Python 3.8+ Miniconda3. The detailed configuration of the simulation experiment is recorded in Table 2.

B. EXPERIMENTAL RESULTS

We choose the heuristic algorithm NodeRank, the baseline algorithm BaseLine based on greedy strategy, and the algorithm R-VNE based on node importance ranking as the comparison algorithms. The performance of the algorithms

TABLE 2. Simulation parameters.

Parameter	Value
Number of satellite nodes	150
Number of microservice service requests	1000
Length of microservice requests	$\mathcal{N}[2, 10]$
Bandwidth resources	$\mathcal{N}[50, 100]$
Bandwidth requirements	$\mathcal{N}[1, 50]$
CPU resources	$\mathcal{N}[50, 100]$
CPU requirements	$\mathcal{N}[1, 50]$

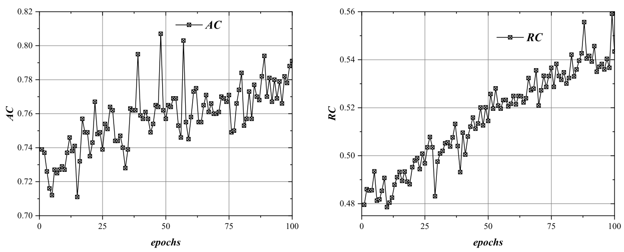


FIGURE 4. Variation curves of parameters related to the training process.

is then compared in three aspects: 1) the acceptance rate of microservice requests and 2) the long-term average gain of deploying microservice requests in satellite nodes and 3) the overall evaluation metrics. The four algorithms including our algorithm are described and summarized in Table 3.

First, we verified the convergence of the algorithm by observing the change curves of the evaluation metric parameters during the training process. We show in Figure 4 the trend of the two metrics, AC and RC, at each training round. At the beginning of the training, the W parameter matrix is randomly initialized, thus the results are poor and fluctuate greatly. As the number of training rounds increases, the three evaluation metrics move in a better direction and achieve better results while stabilizing in the later stages of training. The experimental results also show that our proposed algorithm is convergent and can achieve good results.

For our experiments, the XML network topology files exported using the ALEVIN2 tool are fed into each comparison algorithm. To ensure the reliability and fairness of the experimental results, we use exactly the same environment for all participating algorithms. In addition, since the distribution of random samples is usually fixed, we conduct 100 sets of experiments in each scenario for simulation testing when the number of samples increases in positive correlation with the increase in accuracy. We compared the microservice placement acceptance rate, long benefit-cost ratio, and comprehensive evaluation metrics of the four algorithms. The results are shown in Figure 5, Figure 6, and Figure 7.

Figure 5 shows the comparison of acceptance rates of the four algorithms, and the experimental results can clearly show

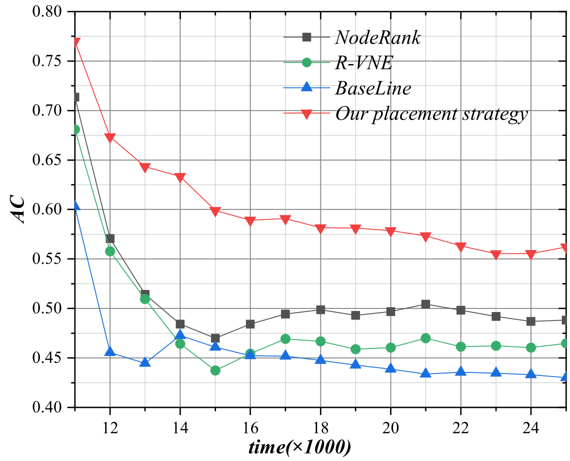


FIGURE 5. Comparison with other algorithms on microservice request acceptance rate.

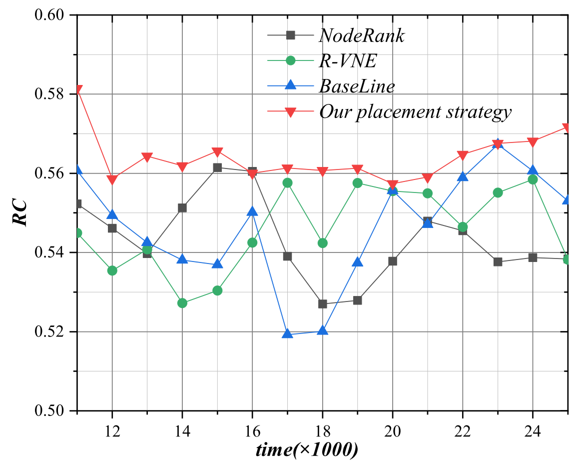


FIGURE 6. Comparison with other algorithms on microservice request long-term revenue-cost ratio.

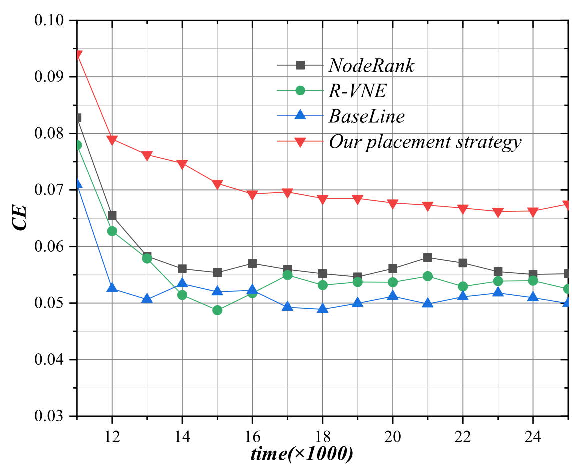


FIGURE 7. Comparison with other algorithms on microservice request comprehensive evaluation indicators.

that the microservice placement scheme proposed in this paper has good performance in terms of microservice request

TABLE 3. Description of the experimental algorithm.

Algorithm	Description
BaseLine	This algorithm uses a greedy algorithm to select the satellite node with the most current remaining resources as the placement node for microservices, and a shortest-circuit algorithm to map the invocation relationships between microservices within a microservice request.
NodeRank [31]	This algorithm first ranks the satellite nodes according to their importance and then uses the BFS policy to map the backward and forward invocation relationships of the microservices in the microservice requests.
R-VNE [32]	This algorithm proposes to use a random rounding-based method to achieve a linear programming relaxation of the MIP corresponding to the microservice placement problem in order to keep the cost of the microservice requests as small as possible.
Our placement stragy	Our strategy uses the attention mechanism to calculate the importance of all satellite networks that can be placed by a microservice node and selects the one with the highest importance for placement.

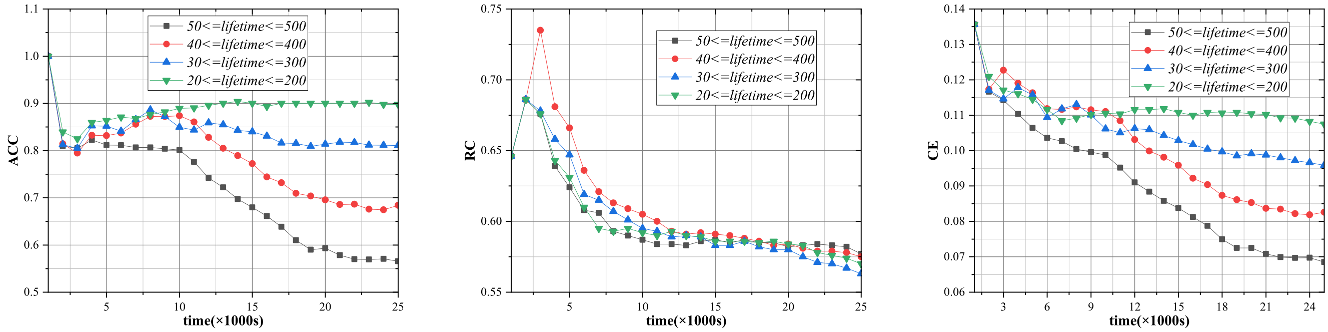


FIGURE 8. The performance of microservice placement policies over time on ACC,RC,CE.

acceptance rates. The comparison algorithm NodeRank computes the importance of satellite nodes considering only the local importance of satellite nodes and is limited by constraint rules, so it leads to a decrease in acceptance rate with the influx of microservice requests and over time. The algorithm R-VNE uses a linear programming approach when processing microservice requests. This algorithm can achieve good results when dealing with small-scale requests, but large-scale service requests often appear simultaneously in real networks, and linear programming cannot handle a problem with a large number of decision variables in a limited computation time, which can reduce the acceptance rate of microservice requests.

Figure 6 shows the performance of the four algorithms in terms of the resource gain overhead ratio. Our proposed scheme also achieves good results in terms of revenue-to-overhead ratio and is more stable than the remaining three placement algorithms. All three algorithms, BaseLine, NodeRank, and R-VNE, prefer to place microservices on the satellite nodes with more resources left, and such placement habits may lead to a decrease in available resources when new microservice requests arrive on the most suitable satellite nodes. The new microservice can be successfully deployed in this satellite node, and the tie revenue will increase, thus the revenue-overhead ratio will increase, and vice versa, the revenue-overhead ratio will decrease. So it causes the fluctuation of the revenue overhead ratio fold of the comparison algorithm in Figure 6. Our proposed algorithm takes into account the available resources of all types of satellite nodes adequately and does not appear to focus only on the node with the most resources left, so the revenue overhead ratio does not fluctuate high and low. The stability exhibited by our proposed algorithm is also exactly what is needed in a real network environment.

As shown in Figure 7, the scheme proposed by us is better than the other three schemes in a comprehensive evaluation. After calculation, our placement algorithm is 18.1%, 25.4%, and 30.1% higher than the other three algorithms in terms of comprehensive evaluation indicators.

Then, we observe the performance of the proposed microservice placement strategy in this paper by setting different microservice service durations. From Figure 8, we can see that when the service duration is longer, the acceptance rate is lower, and the benefit-overhead ratio and comprehensive evaluation index are lower. In the real network environment, with the influx of microservice requests, the microservices placed earlier have not yet completed their missions, so the remaining allocatable resources in the satellite network gradually become less. Our simulation results also show that the trend is very consistent in line with the real situation. Our proposed on-satellite microservice placement strategy can achieve an acceptance rate of more than 55% with a guaranteed revenue-to-overhead ratio. The experimental results demonstrate the effectiveness of our proposed placement strategy, and it has more obvious advantages than manually monitoring and placing microservices.

V. CONCLUSION

With the development of communication technology, the shortcomings of the manual monitoring and management of the microservices approach are gradually revealed. In this paper, we propose a placement strategy for microservices on new satellites in the context of SDN. We model satellite nodes as physical nodes and microservices as virtual nodes and design a microservice placement strategy. The performance of this strategy in terms of service request acceptance rate verifies the stability and effectiveness of our proposed strategy. In the future, we will give more consideration to the impact of security on microservice placement. In addition, we will pay more attention to the impact of dynamic changes in microservices on the network, and research targeted solutions for it.

REFERENCES

[1] J. Liu, Y. Shi, Z. Md. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2714–2741, 4th Quart., 2018.

[2] C. Wang, P. Zhang, N. Kumar, L. Liu, and T. Yang, "GCWCN: 6G-based global coverage wireless communication network architecture," *IEEE Netw.*, vol. 37, no. 3, pp. 218–223, May/Jun. 2023.

[3] W. Qi, Q. Song, X. Wang, L. Guo, and Z. Ning, "SDN-enabled social-aware clustering in 5G-VANET systems," *IEEE Access*, vol. 6, pp. 28213–28224, 2018.

[4] H. K. Janjua, I. de Miguel, R. J. Durán Barroso, M. Masoumi, S. Hosseini, J. C. Aguado, N. Merayo, and E. J. Abril, "A framework for next generation cloud-native SDN cognitive resource orchestrator for IoTs (NG2CRO)," in *Proc. Int. Symp. Distrib. Comput. Artif. Intell.* Cham, Switzerland: Springer, 2023, pp. 399–407.

[5] P. Smet, B. Dhoedt, and P. Simoens, "Docker layer placement for on-demand provisioning of services on edge clouds," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 3, pp. 1161–1174, Sep. 2018.

[6] F. Santos, R. Immich, and E. Madeira, "Multimedia microservice placement in hierarchical multi-tier cloud-to-fog networks," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, May 2021, pp. 1044–1049.

[7] K. Ray, A. Banerjee, and N. C. Narendra, "Proactive microservice placement and migration for mobile edge computing," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Nov. 2020, pp. 28–41.

[8] K. Fu, W. Zhang, Q. Chen, D. Zeng, and M. Guo, "Adaptive resource efficient microservice deployment in cloud-edge continuum," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1825–1840, Aug. 2022.

[9] Y. Duan, Y. Lu, S. Shen, S. Yu, P. Zhang, W. Zhang, and K. K. Igorevich, "NFLCS: An service function chain path optimization strategy based on network-functional layout clustering," *IEEE Trans. Veh. Technol.*, vol. 72, no. 8, pp. 10813–10825, Aug. 2023.

[10] S. Zhang, A. Liu, and X. Liang, "A multi-objective satellite handover strategy based on entropy in LEO satellite communications," in *Proc. IEEE 6th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2020, pp. 723–728.

[11] T. Vergoossen, S. Loarte, R. Bedington, H. Kuiper, and A. Ling, "Modelling of satellite constellations for trusted node QKD networks," *Acta Astronautica*, vol. 173, pp. 164–171, Aug. 2020.

[12] Y. Wang, W. Han, and Z. Nian, "Design of satellite ground management system based on microservices," in *Proc. 3rd Int. Conf. Comput. Sci. Softw. Eng.*, NY, NY, USA, May 2020, pp. 119–123, doi: 10.1145/3403746.3403915.

[13] X. Wang, J. Li, Z. Ning, Q. Song, L. Guo, S. Guo, and M. S. Obaidat, "Wireless powered mobile edge computing networks: A survey," *ACM Comput. Surv.*, vol. 55, pp. 1–37, Jul. 2023, doi: 10.1145/3579992.

[14] A. A. Dowhuszko, J. Fraire, M. Shaat, and A. Pérez-Neira, "LEO satellite constellations to offload optical terrestrial networks in placement of popular content in 5G edge nodes," in *Proc. 22nd Int. Conf. Transparent Opt. Netw. (ICTON)*, Jul. 2020, pp. 1–6.

[15] K. Yin and Q. Du, "On representing resilience requirements of microservice architecture systems," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 31, no. 6, pp. 863–888, Jun. 2021.

[16] P. Zhang, Y. Zhang, N. Kumar, and M. Guizani, "Dynamic SFC embedding algorithm assisted by federated learning in space-air-ground integrated network resource allocation scenario," *IEEE Internet Things J.*, vol. 10, no. 11, pp. 9308–9318, Jun. 2023.

[17] L. Li, L. Qiao, and Q. Chen, "Design and implementation of multi-priority hybrid threshold scheduling algorithm for edge nodes of satellite OBS networks," in *Proc. Int. Conf. Commun., Inf. Syst. Comput. Eng. (CISCE)*, Jul. 2019, pp. 195–198.

[18] X. Wang, Z. Ning, L. Guo, S. Guo, X. Gao, and G. Wang, "Mean-field learning for edge computing in mobile blockchain networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5978–5994, Oct. 2023.

[19] B. Mayer and R. Weinreich, "A dashboard for microservice monitoring and management," in *Proc. IEEE Int. Conf. Softw. Archit. Workshops (ICSAW)*, Apr. 2017, pp. 66–69.

[20] C.-C. Liu, C.-C. Huang, C.-W. Tseng, Y.-T. Yang, and L.-D. Chou, "Service resource management in edge computing based on microservices," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, Aug. 2019, pp. 388–392.

[21] N. Mohammadi and A. Rasoolzadegan, "A pattern-aware design and implementation guideline for microservice-based systems," in *Proc. 27th Int. Comput. Conf., Comput. Soc. Iran (CSICC)*, Feb. 2022, pp. 1–6.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent.*, May 2018, pp. 1–12.

[23] L. Guo, Z. Ning, Q. Song, L. Zhang, and A. Jamalipour, "A QoS-oriented high-efficiency resource allocation scheme in wireless multimedia sensor networks," *IEEE Sensors J.*, vol. 17, no. 5, pp. 1538–1548, Mar. 2017.

[24] P. Jiang, Y. Shen, and Y. Dai, "Efficient software test management system based on microservice architecture," in *Proc. IEEE 10th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC)*, vol. 10, Jun. 2022, pp. 2339–2343.

[25] R. Xu, W. Jin, and D. Kim, "Enhanced service framework based on microservice management and client support provider for efficient user experiment in edge computing environment," *IEEE Access*, vol. 9, pp. 110683–110694, 2021.

[26] D. C. Li, C.-T. Huang, C.-W. Tseng, and L.-D. Chou, "Fuzzy-based microservice resource management platform for edge computing in the Internet of Things," *Sensors*, vol. 21, no. 11, p. 3800, May 2021.

[27] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, May 2015, pp. 98–106.

[28] X. Zhang, Z. Li, C. Lai, and J. Zhang, "Joint edge server placement and service placement in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11261–11274, Jul. 2022.

[29] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes, "Provably efficient algorithms for placement of service function chains with ordering constraints," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 774–782.

[30] H. Zhao, S. Deng, Z. Liu, J. Yin, and S. Dustdar, "Distributed redundancy scheduling for microservice-based applications at the edge," in *Proc. IEEE World Congr. Services (SERVICES)*, 2021, p. 1, doi: 10.1109/SERVICES51467.2021.00012.

[31] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.

[32] M. Chowdhury, M. R. Rahman, and R. Boutaba, ''ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping,'' *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[33] T. N. Kipf and M. Welling, ''Semi-supervised classification with graph convolutional networks,'' 2016, *arXiv:1609.02907*.

[34] J. Liang, Y. Deng, and D. Zeng, ''A deep neural network combined CNN and GCN for remote sensing scene classification,'' *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 4325–4338, 2020.

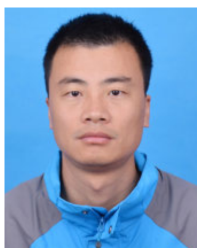
XIANGYU SU graduated from Sun Yat-sen University. He is currently with the Shenzhen Institute of Technology. His main research interests are cloud computing, big data application, and J2EE enterprise application development. His research project ''Private Cloud Storage Application System'' won the first prize of the 12th National Excellent Achievement of Teaching, Teaching and Research Technology Development in Technical Colleges and Universities. The research report ''Cassandra key-value database application research based on cloud computing storage mode'' hosted by him, won the second prize in the 2015 Excellent Scientific Research Achievements of China Vocational Training Association. In addition, he has been awarded an A Grade in the Comprehensive Evaluation of Teaching Quality of Teachers of the College, for many times.

AMR TOLBA (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees from the Department of Mathematics and Computer Science, Faculty of Science, Menoufia University, Egypt, in 2002 and 2006, respectively. He is currently a Full Professor of computer science with King Saud University (KSU), Saudi Arabia. He has authored/coauthored over 130 scientific articles in top-ranked (ISI) international journals, such as IEEE Internet of Things Journal (IoT), *ACM Transactions on Internet Technology* (TOIT), *IEEE Consumer Electronics Magazine* (CEMAG), IEEE Access, IEEE Systems Journal, *Future Generation Computer Systems* (FGCS), *Journal of Network and Computer Applications* (JNCA), *Neural Computing and Applications* (NC&A), *Journal of Ambient Intelligence and Humanized Computing* (JAIHC), *Computer Networks* (COMNET), *Computer Communications* (COMCOM), *P2PNET*, *VCOM*, and *World Wide Web Journal* (WWWJ). He has translated four books into the Arabic language. His main research interests include artificial intelligence (AI), the Internet of Things (IoT), data science, and cloud computing. He has served as a Technical Program Committee (TPC) Member for several conferences, such as DSIT 2022, CICA 2022, EAI MobiHealth 2021, DSS 2021, AEMCSE 2021, ICBDM 2021, ICISE 2021, DSS 2020, NCO 2020, ICISE 2019, ICCSEA 2019, DSS 2019, FCES 19, ICISE 2018, ESG 2018, Smart Data 2017, NECO 2017, NC 2017, WEMNET 2017, NET 2017, and Smart Data 2016. He has been included in the list of the top 2% of influential researchers globally (prepared by scientists from Stanford University, USA), in 2020, 2021, and 2022. He served as an associate editor/a guest editor for several ISI journals.

YUXI LU is currently pursuing the master's degree in computer science and technology with the School of Computer Science and Technology, China University of Petroleum (East China). His research interests include network virtualization and network artificial intelligence.

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

JIAN WANG (Senior Member, IEEE) received the B.S. degree in computational mathematics from the China University of Petroleum (East China), Qingdao, China, in 2002, and the M.S. and Ph.D. degrees in computational mathematics from the Dalian University of Technology, Dalian, China, in 2008 and 2012, respectively. He is currently a Professor and the Head of the Laboratory for Intelligent Information Processing, College of Science, China University of Petroleum (East China). He received several grants from the National Science Foundation of China, the National Key Research and Development Program of China, the Natural Science Foundation of Shandong Province, and the Fundamental Research Funds for the Central Universities. His research interests include computational intelligence, machine learning, pattern recognition, deep learning, differential programming, clustering, fuzzy systems, and evolutionary computation. He has served as the General Chair, the Program Chair, and the Co-Program Chair for several conferences, such as the International Symposium on New Trends in Computational Intelligence, IEEE Symposium Series on Computational Intelligence, and International Symposium on Neural Networks. He serves as an Associate Editor for the IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Emerging Topics in Computational Intelligence, *Information Sciences*, *International Journal of Machine Learning and Cybernetics*, and *Journal of Applied Computer Science Methods*. He also serves on the editorial board for the *Neural Computing and Applications* and *Complex and Intelligent Systems*. He served as a Guest Editor for the *Neural Computing and Applications and Computational Intelligence*.



PEIYING ZHANG (Member, IEEE) received the Ph.D. degree from the School of Information and Communication Engineering, University of Beijing University of Posts and Telecommunications, in 2019. He is currently an Associate Professor with the College of Computer Science and Technology, China University of Petroleum (East China). He has published multiple IEEE/ACM Transactions/journal/magazine articles, such as IEEE Transactions on Industrial Informatics, IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Vehicular Technology, IEEE Transactions on Network Science and Engineering, IEEE Transactions on Network and Service Management, IEEE Transactions on Emerging Topics in Computing, IEEE Network, IEEE Internet of Things Journal, *ACM TALLIP*, *Computer Communications*, and *IEEE Communications Magazine*. His research interests include semantic computing, future internet architecture, network virtualization, and artificial intelligence for networking. He served as a Technical Program Committee Member for IEEE ICC3, IEEE ICC22, DPPR 2021, ISCIT 2016, ISCIT 2017, ISCIT 2018, ISCIT 2019, GLOBECOM 2022, GLOBECOM 2021, GLOBECOM 2019, COMNETSAT 2020, ICICoS 2022, SoftIoT 2021, IWCMC-Satellite 2019, IWCMC-Satellite 2020, and IWCMC-Satellite 2022. He is the Leading Guest Editor of *Electronics* and *Frontiers in Psychiatry* and an Editorial Board Member of *Artificial Intelligence and Applications* (AIA).