Dynamic resource allocation for virtual network function placement in satellite edge clouds

Xiangqiang Gao, Rongke Liu, Aryan Kaushik, Hangyu Zhang

Publication date
10-06-2023

Document Version
Accepted version

Published in
IEEE Transactions on Network Science and Engineering

Link to external publisher version
https://doi.org/10.1109/TNSE.2022.3159796

Dynamic Resource Allocation for Virtual Network Function Placement in Satellite Edge Clouds

Xiangqiang Gao, Rongke Liu, *Senior Member, IEEE,* Aryan Kaushik, *Member, IEEE,* and Hangyu Zhang

***Abstract*—Satellite edge computing has become a promising way to provide computing services for Internet of Things (IoT) users in remote areas, which are out of the coverage of terrestrial networks. Nevertheless, it is not suitable for large-scale IoT users due to the resource limitation of satellites. Cloud computing can provide sufficient available resources for IoT users, but it does not meet delay-sensitive services as high network latency. Satellite edge clouds can facilitate flexible service provisioning for numerous IoT users by incorporating the advantages of edge computing and cloud computing. In this paper, we investigate the dynamic resource allocation problem for virtual network function (VNF) placement in satellite edge clouds. The aim is to minimize the network bandwidth cost and the service end-to-end delay jointly. We formulate the VNF placement problem as an integer non-linear programming problem and then propose a distributed VNF placement (D-VNFP) algorithm to address it. The experiments are conducted to evaluate the performance of the proposed D-VNFP algorithm, where Viterbi and Game theory are considered as the baseline algorithms. The results show that the proposed D-VNFP algorithm is effective and efficient for solving the VNF placement problem in satellite edge clouds.**

***Index Terms*—Satellite edge clouds, resource allocation, virtual network function (VNF) placement, network bandwidth cost, service end-to-end delay, distributed algorithm.**

I. Introduction

THE Internet of Things (IoT) has been widely applied in various fields, e.g., disaster monitoring, ocean transportation, target recognition and tracking, etc [1]. Large amount of requested services produced by IoT users need to be offloaded by terrestrial networks to remote servers for execution as IoT users have low computing capacity and battery power. However, terrestrial networks have not been set up in some remote areas due to high capital expenditures and harsh environments. Fortunately, low earth orbit (LEO) satellite networks, which have low communication delay and global coverage, can provide data collection, computing, and communication services for remote IoT users without the coverage of terrestrial networks [2].

Satellite edge computing has been considered as a new paradigm to provide computing services near user sides [3]. We can deploy edge servers on LEO satellites and provide satellite edge computing services for remote IoT users, which can improve the real-time performance of computing service significantly [4], [5]. However, LEO satellites have the resource limitation of computing, storage, and bandwidth due to satellite physical constraints (e.g., satellite volume, energy, weight, etc.) [6]. Cloud computing can provide sufficient available resources for remote IoT users [7], [8], where data centers are usually located far away from IoT users. When requested services from IoT users are offloaded to cloud servers by satellite networks, it will lead to high network latency as well as heavy network load, which can degrade their processing performance [9], [10]. Therefore, collaborative edge clouds [11], which combine the advantages of edge computing and cloud computing, can provide edge computing services for delay-sensitive users and offload compute-intensive services to data centers for further processing [12], [13].

Network function virtualization (NFV) can facilitate the decoupling of software and hardware and replace dedicated hardware middleboxes with software modules to run on commercial servers [14]. In that case, the requested service for an IoT user can be indicated as a service function chain (SFC), which consists of a sequence-ordered set of virtual network functions (VNFs) [15]. Collaborative edge clouds combining with NFV have been a promising approach for service provisioning and resource management in a flexible and efficient way [16].

Extensive research has been conducted on the VNF placement problem in edge clouds, where the aims are to optimize network bandwidth consumption [17], network throughput [18], maximum link load ratio [19], etc. Note that most of the existing work [17]–[19] focuses on placing VNFs for IoT users to optimize the goal from the perspective of either the users or the service providers. However, the VNF placement strategy considered from the perspective of either the users or the service providers may degrade the benefits related to the other side. Therefore, it should be more reasonable for the VNF placement strategy to optimize the objectives of users and service providers jointly [20], especially in the scenarios with severe resource constraints and service-sensitive requirements, such as satellite edge computing. Compared with ground edge clouds, the available resources of satellite networks are severely limited, e.g., computing, storage, and bandwidth [21]. Moreover, the satellite network topology is varying over time due to the mobility of satellites, which brings a new dimension and difficulty in solving the VNF placement problem [22].

As the periodical and dynamic characteristics of satellites, a satellite network can be divided based on a time slot method

This work was supported by Beijing Natural Science Foundation (L202003).

X. Gao and H. Zhang are with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: {xggao, zhanghangyu}@buaa.edu.cn).

R. Liu is with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China and with Shenzhen Institute of Beihang University, Shenzhen 518038, China (e-mail: rongke_liu@buaa.edu.cn).

A. Kaushik is with the School of Engineering and Informatics, University of Sussex, Brighton BN1 9RH, United Kingdom (e-mail: aryan.kaushik@sussex.ac.uk).

in a quasi-static scenario. Then the VNF placement problem in satellite edge computing is discussed based on potential game [23] to maximize the network utility in our previous work [24]. However, the satellite coverage for IoT users as well as collaborative computing service in satellite edge clouds is not considered in that work. Actually, IoT users are in the coverage of specific satellites and can only directly communicate with the satellites covering them.

In this paper, we investigate the VNF placement problem in satellite edge clouds. We formulate the VNF placement problem as an integer non-linear programming problem and our aim is to minimize the service end-to-end delay and the network bandwidth cost jointly. Furthermore, we consider that an IoT user is covered by multiple satellites, where different VNF placement strategies have an influence on the performance in terms of network bandwidth cost and service end-to-end delay. As the VNF placement problem is NP-hard [25], [26], we propose a distributed virtual network function placement (D-VNFP) algorithm to address the problem. Each satellite is viewed as an agent and can make the VNF placement strategies for user requests independently. Note that all satellites make the VNF placement strategies for user requests in parallel. Due to the resource limitation of satellite networks, there exists a potential resource conflict, which can be solved by competing available network resources with each other in a priority-based way. The main contributions are summarized as follows.

- We study the dynamic resource allocation problem for VNF placement in satellite edge clouds.
- We formulate the VNF placement problem in satellite edge clouds and analyze the problem complexity.
- We propose a distributed virtual network function placement algorithm to tackle the problem. We conduct extensive simulations to evaluate the performance of the proposed D-VNFP algorithm compared with Viterbi and Game theory.

The remainder of this paper is organized as follows. Section II briefly reviews related work concerning VNF placement in edge clouds. Section III introduces the system model in satellite edge clouds. We formulate the VNF placement problem in Section IV. We propose a distributed virtual network function placement algorithm to address the VNF placement problem in Section V. In Section VI, we evaluate the performance of the proposed D-VNFP algorithm. Finally, we provide the conclusion of this paper in Section VII.

II. Related Work

NFV can facilitate resource allocation on-demand and agile service provisioning in satellite networks [27], [28], therefore NFV-enabled satellite networks have been investigated for managing dynamic network resources in the recent literature. In [29], the authors considered a satellite network as the software defined time-evolving graph and discussed the VNF-based service provisioning problem while minimizing the resource consumption of satellite-to-satellite links. The authors in [30] studied the SFC deployment problem to minimize the service end-to-end delay in satellite networks. In a multi-domain NFV-enabled satellite-terrestrial networks, the authors in [31] proposed a horizontal-based multi-domain SFC orchestration to improve the service end-to-end delay. However, the related work just discussed the VNF placement problem on behalf of either the users or the service providers. Moreover, function orchestrators and network controllers are responsible for managing and allocating satellite network resources. In that case, satellites are non-autonomous to provide computing services for users passively by the VNF placement strategies. In our work, we discuss the VNF placement problem to optimize the service end-to-end delay and the network bandwidth cost jointly from the perspective of both the users and the service providers. We also consider each satellite as an autonomous agent, which can manage network resources and make the VNF placement strategy autonomously.

Most of the existing work focuses on the VNF placement in conventional edge clouds. For example, the authors in [32] studied a resource allocation scheme based on context-aware grouping for VNF placement in fifth generation edge networks. The authors in [33] investigated the VNF services for mobile users in mobile edge cloud networks. The authors in [34] discussed the service provisioning for VNF-enabled IoT users in mobile edge clouds. Note that the existing related work in conventional edge clouds considers that edge cloud networks are usually fixed and cannot be varying over time. However, satellite networks are dynamic and time-varying, which can bring a new challenge for VNF placement in satellite edge clouds.

Moreover, for VNF placement and resource management, centralized approaches lack scalability and flexibility as the increase in IoT users and network scales. Therefore, some researchers focus on decentralized approaches to provide computing services for users in a scalable and flexible manner. The authors in [35] investigated a distributed approach based on alternating direction method of multipliers for VNF placement in large-scale networks. The authors in [36] discussed the resource allocation for NFV by joint benders decomposition and alternating direction method of multipliers. Furthermore, as the self-interested and competitive characteristics of players in non-cooperative game, non-cooperative game approaches are widely used to solve resource allocation in a distributed way. In [37], [38], the authors proposed weighted congestion game approaches to manage network resources for VNF placement, respectively. A VNF placement approach based on potential game is also presented to maximize the total network payoff in [24]. However, different from the above approaches, we propose a distributed VNF placement approach based on multi-agent systems to manage network resources in satellite edge clouds. All agents can provide computing services for users in parallel and compete available resources with each other in a priority-based way. Compared with non-cooperative game, the proposed D-VNFP algorithm has a tradeoff between strategy quality and computation complexity.

III. System Model

In this section, we firstly provide the hierarchical architecture of computing services for IoT users in satellite edge clouds. Then we discuss the user request model and the VNF

TABLE I
LIST OF SYMBOLS.

Satellite Network	
$G(V, E)$	The satellite network with satellites V and links E.
R	The set of resources offered by satellite v, $v \in V$.
C_v^r	The r-th resource capacity of satellite v, $r \in R$.
B_e, t_e	Bandwidth and delay of link e, $e \in E$.
dc	The cloud data center.
$B_{v,dc}$	Bandwidth capacity of the link between v and dc.
$P_{v_1}^{v_2}$	Set of the d shortest paths between v_1 and v_2.
User Requests	
U	The set of M user requests.
F_u, H_u	VNFs F_u and edges H_u of user request u.
$f_{u,i}$	The i-th VNF of user request u.
s_u, d_u	Source and destination of user request u.
$c_{u,i}^r$	The r-th resource requirements of $f_{u,i}$.
$t_{u,i}$	The execution time of $f_{u,i}$.
$h_u^{i_1,i_2}$	Edge between f_{u,i_1} and f_{u,i_2}.
$b_u^{i_1,i_2}$	The bandwidth requirements of edge $h_u^{i_1,i_2}$.
t_u^{max}	Maximum acceptable delay of user request u.
Binary Variables	
x_u	$x_u = 1$ if u is deployed to satellite edge clouds.
y_u	$y_u = 1$ if satellites provide computing service for u.
$z_{u,i}^v$	$z_{u,i}^v = 1$ if $f_{u,i}$ is deployed to satellite v.
$w_{u,p}^{i_1,i_2}$	$w_{u,p}^{i_1,i_2} = 1$ if path p is used by $h_u^{i_1,i_2}$.
w_e^p	$w_e^p = 1$ if link e is used by path p.
Objective Variables	
C_{bw}	Average satellite network bandwidth cost.
C_{user}	Average service end-to-end delay.
C_{total}	Total deployment cost.

placement problem in satellite edge clouds. The main symbols concerning the system model are summarized in Table I.

A. Satellite Edge Cloud Networks

For the VNF placement problem in satellite edge clouds, we consider a hierarchical satellite edge cloud architecture including IoT users, LEO satellites, and a cloud data center, as shown in Fig. 1. IoT users with low computing capacity and battery power, are distributed to offer environment observation and local computing in remote areas without the coverage of terrestrial networks. Therefore, the requests from IoT users can be preferentially offloaded to satellites and the data center for further processing. Edge servers on satellites can provide edge computing services for delay-sensitive user requests. The computing-intensive user requests can be offloaded by satellites networks to cloud servers, which can provide sufficient resource provisioning for users but there also exists high processing delay due to the characteristics of satellite networks. Note that user requests may fail to deploy as they are not in the coverage of satellites or their resource requirements can not be satisfied. In that case, these user requests need to be performed locally [23].

We denote the satellite network as a graph $G(V, E)$, where the parameters V and E represent the set of satellites and the set of inter-satellite links, respectively. The number of satellites in the set V is N. For satellite $v, \forall v \in V$, the set of resource types is indicated as $R = \{CPU, memory\}$ and the r-th resource capacity is expressed by $C_v^r, \forall r \in R$. There are four inter-satellite links for a satellite, in which two links

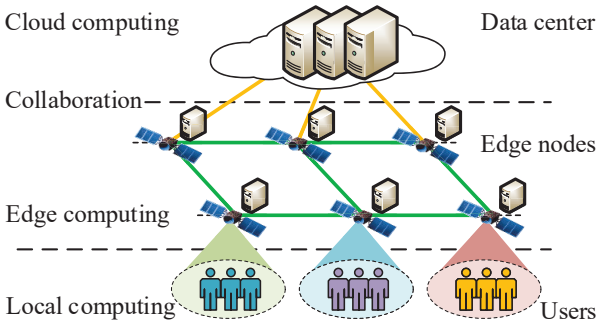


Fig. 1. Hierarchical satellite edge cloud architecture.

are from adjacent satellites in the same orbit and the other two links are from inter-orbit adjacent satellites [39]. For link $e, \forall e \in E$, we indicate the bandwidth capacity as B_e and the transmission delay as t_e, respectively. We assume that there are sufficient resources for user requests in cloud data center dc. We use $e_{v,dc}$ to indicate the link between satellite v and cloud data center dc, $B_{v,dc}$ to indicate the bandwidth capacity, and $t_{v,dc}$ to indicate the transmission delay. We also assume that the cloud data center is in the coverage of satellites and can communicate with the satellites by the satellite-ground link.

B. User Requests

The set of user requests is represented as U, where the number of user requests is M. User request $u, \forall u \in U$, can be considered as a directed graph $G(F_u, H_u)$. For user request u, we use F_u to indicate the set of VNFs, where VNF $f_{u,i}$, $f_{u,i} \in F_u$, represents the i-th VNF. VNF $f_{u,1} = f_{u,s}$ represents the source function and VNF $f_{u,|F_u|} = f_{u,d}$ represents the destination function, respectively. We indicate source s_u as a sensor to collect the data and destination d_u as an actuator to interact with the environment by the processing results, where s_u and d_u can be different but known in advance. All the VNFs except $f_{u,s}$ and $f_{u,d}$ have the requirements of CPU, memory, and execution time. We denote the resource requirements for VNF $f_{u,i}$ as $c_{u,i}^r, \forall r \in R$, and the execution time as $t_{u,i}$, respectively. We also use H_u to indicate the set of edges. The edge between the adjacent VNFs f_{u,i_1} and f_{u,i_2} is indicated as $h_u^{i_1,i_2}$, where the corresponding bandwidth requirements can be indicated as $b_u^{i_1,i_2}$. The maximum acceptable delay is t_u^{max}. Supposing that all IoT users are distributed in remote areas that are out of the coverage of terrestrial networks, only these IoT users in the coverage of satellites can be provided computing services by satellite edge clouds. Satellites can only directly communicate with IoT users in the coverage. That is, IoT users without the coverage of satellites will perform their requested services locally. Moreover, the VNFs cannot be shared among different user requests.

C. VNF Placement in Satellite Edge Clouds

Similar to the existing related work in [22], [23], we consider the VNF placement problem in a quasi-static scenario. The satellite network topology remains unchanged during each time slot and can vary over different time slots [22].

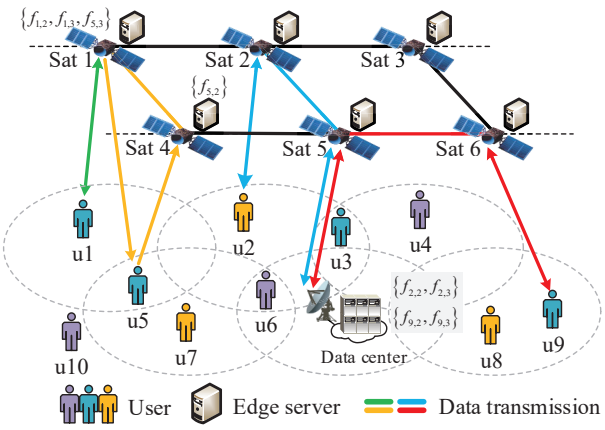


Fig. 2. Example of deploying user requests in a satellite edge cloud network.

User requests can randomly occur and end in each time slot. However, the VNF placement strategy for each user request remains unchanged and the service requirements can be guaranteed during the running time [23]. All available satellite network resources are virtualized to provide service provisioning for user requests on-demand. Then we use a batch processing model to deploy user requests to satellite edge clouds in dynamic environment.

At the beginning of each time slot, we assume that several user requests arrive and will be provided computing services in the next time slot. We firstly update the current satellite network state as per the availability of network resources, where the network resources used by the completed user requests in the previous time slot can be freed to be available resources for new user requests in the next time slot. Then we collect the new user requests to appear in the next time slot and allocate the available resources by the proposed D-VNFP algorithm for them on-demand. To more clearly describe the VNF placement problem, we define the neighbouring satellites and the access satellite for each user request, respectively.

Definition 1 *(**Neighbouring Satellite**) For user request u, when the source is in the coverage of satellites $N(u,s)$, then we consider that the satellites $N(u,s)$ are the source neighbouring satellites. Similarly, we define the neighbouring satellites $N(u,d)$ for the destination and $N(dc)$ for the cloud data center, respectively.*

Definition 2 *(**Access Satellite**) When a user request requests computing service from satellite edge clouds, the access information concerning the user request will be firstly sent to one of its source neighbouring satellites. the source neighbouring satellite can make the VNF placement strategy for the user request. In that case, the source neighbouring satellite is viewed as the access satellite for the source of the user request. Similarly, we also define the access satellites for the destination of the user request and the cloud data center, respectively.*

As IoT users have low computing capacity and battery power, the requested service information of each user request will be sent to the access satellite of the source. Then the source access satellite is responsible for making the VNF placement strategy by the current satellite network state and the resource requirements.

For each user request, the VNF placement strategy will be shared among the related satellites in order to build the SFC. VNFs $f_{u,s}$ and $f_{u,d}$ are deployed on the corresponding source and destination neighbouring satellites, which can offload the requested service to satellite edge clouds and forward the processing results to the destination, respectively. Note that different VNF placement strategies have an influence on the network bandwidth cost and the service end-to-end delay.

Fig. 2 shows an example of deploying user requests in a satellite edge cloud network. There are 6 satellites and each satellite can provide computing services for user requests. The data flow from user requests can be routed between satellites via inter-satellite links. There are also 10 user requests and a data center on the ground, where the data center is in the coverage of *Sat5*. User request u_1 is in the coverage of *Sat1* and there are sufficient available resources for *Sat1*. Considering that a user request can just communicate with its neighbouring satellites, VNFs $f_{1,s}$, $f_{1,2}$, $f_{1,3}$, and $f_{1,d}$ for user request u_1 will be deployed to *Sat1*. User request u_5 is in the coverage of *Sat1* and *Sat4*, where *Sat4* is the access satellite for the source of user request u_5. Due to the resource limitation of *Sat4*, we deploy VNFs $f_{5,s}$ and $f_{5,2}$ to *Sat4* and VNFs $f_{5,3}$ and $f_{5,d}$ to *Sat1*, respectively. In that case, user request u_5 will be offloaded to the satellite network by *Sat4*. *Sat1* is the access satellite for the destination of user request u_5, therefore, the processing results will be sent back to user request u_5 by *Sat1*. When the available resources of satellites are insufficient, the data center will provide computing services for user requests. For example, user request u_9 can be offloaded to the data center by *Sat6* and *Sat5*, user request u_2 can be offloaded to the data center by *Sat2* and *Sat5*. Then VNFs $f_{2,2}$, $f_{2,3}$, $f_{9,2}$, and $f_{9,3}$ will be executed in the data center. Note that VNFs $f_{2,s}$ and $f_{2,d}$ are deployed to *Sat2* and VNFs $f_{9,s}$ and $f_{9,d}$ are deployed to *Sat6*, respectively. User request u_{10} without the coverage of any satellites will be performed locally.

IV. VNF Placement Problem

A. Variable Definitions

In this paper, we formulate the VNF placement problem to minimize the network bandwidth cost and the service end-to-end delay jointly in satellite edge clouds, where the main variables related to the VNF placement problem are also summarized in Table I.

We use a binary variable $x_u = \{0,1\}$ to indicate whether user request u is deployed to satellites and the data center. If user request u is deployed, $x_u = 1$, otherwise $x_u = 0$.

When user request u is successfully deployed, we use a binary variable $y_u = \{0,1\}$ to indicate whether the user request is deployed to satellites or the data center. If the user request is deployed to satellites, $y_u = 1$, otherwise $y_u = 0$.

If user request u is deployed to satellites, we use a binary variable $z_{u,i}^v = \{0,1\}$ to indicate whether VNF $f_{u,i}$ is deployed to satellite v. If VNF $f_{u,i}$ is deployed to satellite v, $z_{u,i}^v = 1$, otherwise $z_{u,i}^v = 0$.

We assume that the set of the d shortest paths between any two satellites v_1 and v_2 can be indicated as $P_{v_1}^{v_2}$. We use a binary variable $w_{u,p}^{i_1,i_2} = \{0,1\}$ to indicate whether path p is used by edge $h_u^{i_1,i_2}$. If path p is used by edge $h_u^{i_1,i_2}$, $w_{u,p}^{i_1,i_2}$=1, otherwise $w_{u,p}^{i_1,i_2} = 0$.

For path p, we use a binary variable $w_e^p = \{0,1\}$ to indicate whether link e is used by path p. If link e is used by path p, $w_e^p = 1$, otherwise $w_e^p = 0$.

The source of a user request can only communicate with its neighbouring satellites. We assume that the source of user request u is in the coverage of satellite v, that is

$$s_u \in cov(v), \forall u \in U, \forall v \in V, \tag{1}$$

where we indicate $cov(v)$ as the coverage of satellite v. Then the set $N(u,s)$ of neighbouring satellites for the source of user request u can be described as

$$N(u,s) = \{v |\, s_u \in cov(v)\}, \forall v \in V. \tag{2}$$

Similarly, the set $N(u,d)$ of neighbouring satellites for the destination of user request u can be described as

$$N(u,d) = \{v |\, d_u \in cov(v)\}, \forall v \in V, \tag{3}$$

the set of neighbouring satellites for data center dc is indicated as

$$N(dc) = \{v |\, dc \in cov(v)\}, \forall v \in V. \tag{4}$$

When user requests are deployed to satellite edge clouds, their neighbouring satellites for user requests and the data center will be considered as access satellites to offload requested services and receive the processing results. Therefore, we need to specify which one neighbouring satellite is used to be an access satellite.

We use a binary variable $q_{u,s}^v = \{0,1\}$ to indicate whether source neighbouring satellite v is an access node for user request u to offload the requested service. If neighbouring satellite v is an access node for user request u to offload the requested service, $q_{u,s}^v = 1$, otherwise $q_{u,s}^n = 0$. Similarly, another binary variable $q_{u,d}^v = \{0,1\}$ is used to indicate whether destination neighbouring satellite v is an access node for user request u to obtain the processing results. If neighbouring satellite v is an access node for user request u to obtain the processing results, $q_{u,d}^v = 1$, otherwise $q_{u,d}^v = 0$.

For cloud computing, the satellite network is just considered as a communication network and can not provide computing services for user requests, all requested services will be offloaded to the data center for further processing. We use a binary variable $q_{u,s}^{v,dc} = \{0,1\}$ to indicate whether neighbouring satellite v of the data center is an access node for user request u to offload the requested service to the data center. If neighbouring satellite v of the data center is an access node for user request u to offload the requested service to the data center, $q_{u,s}^{v,dc} = 1$, otherwise $q_{u,s}^{v,dc} = 0$. We use another binary variable $q_{u,d}^{v,dc} = \{0,1\}$ to indicate whether neighbouring satellite v of the data center is an access node for user request u to send back the processing results. If neighbouring satellite v of the data center is an access node for user request u to send back the processing results, $q_{u,d}^{v,dc} = 1$, otherwise $q_{u,d}^{v,dc} = 0$.

B. Problem Formulation

When we deploy user requests to satellite edge clouds, multiple physical constraints should be considered, such as network resource capacity, service quality, and deployment implementation.

When user request u is offloaded to satellite edge clouds, we need to guarantee that only one source and destination neighbouring satellites can be used. For $\forall u \in U, \forall v_1 \in N(u,s), \forall v_2 \in N(u,d)$, the access constraint of neighbouring satellites can be represented as

$$x_u \cdot (1 - \sum_{v_1} q_{u,s}^{v_1} \cdot \sum_{v_2} q_{u,d}^{v_2}) = 0. \tag{5}$$

When user request u is deployed to satellites, each VNF $f_{u,i}$ can be deployed to only one satellite. For $\forall u \in U$, the VNF placement constraint can be indicated as

$$x_u \cdot y_u \cdot (1 - \sum_{v} z_{u,i}^v) = 0, \forall f_{u,i} \in F_u, \forall v \in V. \tag{6}$$

For user request u, when two adjacent VNFs f_{u,i_1} and f_{u,i_2} are deployed to satellites v_1 and v_2, respectively, it needs to be guaranteed that one of the d shortest paths between satellites v_1 and v_2 can be used to route the traffic flow from VNF f_{u,i_1} to VNF f_{u,i_2}. The path selection constraint for $\forall v_1, v_2 \in V, \forall u \in U, \forall p \in P_{v_1}^{v_2}$ can be expressed as

$$x_u \cdot y_u \cdot \sum_{v_1,v_2} (z_{u,i_1}^{v_1} \cdot z_{u,i_2}^{v_2} - \sum_{p} w_{u,p}^{i_1,i_2}) = 0. \tag{7}$$

As each satellite has the limited resource capacity in terms of CPU and memory, the resource requirements of user requests for each satellite should not be greater than the available resources. The r-th used resources of satellite v is indicated as $c_{v,used}^r$. For $\forall v \in V, \forall r \in R, \forall u \in U, \forall f_{u,i} \in F_u$, we can indicate the satellite resource constraint as

$$c_{v,used}^r + \sum_{u} x_u \cdot y_u \sum_{f_{u,i}} z_{u,i}^v \cdot c_{u,i}^r \le C_v^r. \tag{8}$$

Furthermore, we need to guarantee that the bandwidth requirements of user requests should not exceed the available bandwidth resources. When user requests are deployed to satellites, for $\forall u \in U$, $\forall h_u^{i_1,i_2} \in H_u, \forall v_1, v_2 \in V, \forall p \in P_{v_1}^{v_2}$, the bandwidth requirements $b_{e,1}$ of link e can be indicated as

$$b_{e,1} = \sum_{u} x_u \cdot y_u \sum_{h_u^{i_1,i_2}} \sum_{v_1,v_2} z_{u,i_1}^{v_1} \cdot z_{u,i_2}^{v_2} \sum_{p} w_{u,p}^{i_1,i_2} \cdot w_e^p \cdot b_u^{i_1,i_2}. \tag{9}$$

When the available resources of satellites are insufficient, we will consider to offload user requests to the data center for further processing while guaranteeing their service quality. To simplify the proposed VNF placement problem, we assume that all the VNFs of a user request except the source function and the destination function will be offloaded to the data center when a user request needs to be offloaded to the data center. Specifically, the satellite network is responsible for routing the data flow from the source to the data center and sending the processing results back from the data center to the destination. When requested services for user requests are offloaded to the

data center, the bandwidth requirements of link e for $\forall u \in U, \forall v_1 \in N(u,s), \forall v_2 \in N(dc), \forall p \in P_{v_1}^{v_2}$ can be indicated as

$$b_{e,2}^1 = \sum_{v_1,v_2} q_{u,s}^{v_1} \cdot q_{u,s}^{v_2,dc} \sum_p w_{u,p}^{s,2} \cdot w_e^p \cdot b_u^{s,2}. \quad (10)$$

When the processing results of user requests are sent back from the data center to their destinations, the bandwidth requirements of link e for $\forall u \in U, \forall v_1 \in N(dc), \forall v_2 \in N(u,d), \forall p \in P_{v_1}^{v_2}$ can be indicated as

$$b_{e,2}^2 = \sum_{v_1,v_2} q_{u,d}^{v_1,dc} \cdot q_{u,d}^{v_2} \sum_p w_{u,p}^{|F_u|-1,d} \cdot w_e^p \cdot b_u^{|F_u|-1,d}. \quad (11)$$

Then the bandwidth requirements of link e for user requests, which are deployed to cloud servers, can be indicated as

$$b_{e,2} = \sum_u x_u \cdot (1-y_u) \cdot (b_{e,2}^1 + b_{e,2}^2), \forall u \in U. \quad (12)$$

We denote the used bandwidth resources of link e as $b_{e,used}$, then the bandwidth resource constraint of link e can be represented as

$$b_{e,used} + b_{e,1} + b_{e,2} \le B_e, \forall e \in E. \quad (13)$$

When user request u is deployed to cloud servers, we need to guarantee that there exists available neighbouring satellites for the data center to offload the requested service and receive the processing results. The neighbouring satellite constraint for $\forall u \in U, \forall v_1, v_2 \in N(dc)$ can be described as

$$x_u \cdot (1-y_u) \cdot (1 - \sum_{v_1} q_{u,s}^{v_1,dc} \cdot \sum_{v_2} q_{u,d}^{v_2,dc}) = 0. \quad (14)$$

We also ensure that there exists an available path between the neighbouring satellites for user request u and the data center. When the requested service is offloaded to the cloud center, the path select constraint for $\forall u \in U, \forall v_1 \in N(u,s), \forall v_2 \in N(dc), \forall p \in P_{v_1}^{v_2}$ can be described as

$$x_u \cdot (1-y_u) \cdot \sum_{v_1,v_2} (q_{u,s}^{v_1} \cdot q_{u,d}^{v_2,dc} - \sum_p w_{u,p}^{s,2}) = 0. \quad (15)$$

When the processing results are sent back to the destination, the path select constraint for $\forall u \in U, \forall v_1 \in N(dc), \forall v_2 \in N(u,d), \forall p \in P_{v_1}^{v_2}$ can be described as

$$x_u \cdot (1-y_u) \cdot \sum_{v_1,v_2} (q_{u,d}^{v_1,dc} \cdot q_{u,d}^{v_2} - \sum_p w_{u,p}^{|F_u|-1,d}) = 0. \quad (16)$$

Furthermore, the bandwidth requirements between the data center and the neighbouring satellite $v, v \in N(dc)$, should not be more than the available bandwidth resources. We denote the used bandwidth resources of the link as $b_{v,dc}^{used}$. The bandwidth resource constraint for link $e_{v,dc}$ can be indicated as

$$b_{v,dc}^{used} + b_{v,dc}^{req} \le B_{v,dc}, \quad (17)$$

where $b_{v,dc}^{req}$ denotes the bandwidth requirements of user requests and can be indicated for $\forall u \in U, \forall v \in N(dc)$ as

$$b_{v,dc}^{req} = \sum_u x_u \cdot (1-y_u) \cdot (b_u^{s,2} \cdot q_{u,s}^{v,dc} + b_u^{|F_u|-1,d} \cdot q_{u,d}^{v,dc}). \quad (18)$$

We also need to ensure that the processing delay of a user request, which is the sum of execution delay and transmission delay, can not exceed the maximum acceptable delay. The execution delay of user request u can be described as

$$t_u^{cmp} = \sum_{f_{u,i}} t_{u,i}, \forall u \in U, \forall f_{u,i} \in F_u. \quad (19)$$

We denote the transmission delay from the source to the source neighbouring satellite $\bar{v}_1$ as $t_{s,\bar{v}_1}$ and from the destination neighbouring satellite $\bar{v}_2$ to the destination as $t_{\bar{v}_2,d}$, respectively. When user request u is deployed to satellites, for $\forall \bar{v}_1 \in N(u,s), \forall \bar{v}_2 \in N(u,d)$, the transmission delay $t_{u,1}^{tr}$ can be indicated as

$$t_{u,1}^{tr} = \sum_{\bar{v}_1} q_{u,s}^{\bar{v}_1} \cdot t_{s,\bar{v}_1} + t_{u,1}^{sat} + \sum_{\bar{v}_2} q_{u,d}^{\bar{v}_2} \cdot t_{\bar{v}_2,d}, \quad (20)$$

where the transmission delay of inter-satellite links is denoted as $t_{u,1}^{sat}$. For $\forall h_u^{i_1,i_2} \in H_u$, $\forall p \in P_{v_1}^{v_2}, \forall v_1, v_2 \in V, \forall e \in p$, $t_{u,1}^{sat}$ can be described as

$$t_{u,1}^{sat} = \sum_{h_u^{i_1,i_2}} \sum_{v_1,v_2} z_{u,i_1}^{v_1} \cdot z_{u,i_2}^{v_2} \sum_p \sum_e w_{u,p}^{i_1,i_2} \cdot t_e. \quad (21)$$

In our work, we just consider the execution delay of the VNFs for user requests and neglect the processing delay of switches and links in the data center. We assume that the transmission delay between the data center and the neighbouring satellite v_1 for computation offloading is $t_{v_1,dc}$ and the transmission delay between the data center and the neighbouring satellite v_2 for receiving the processing results is t_{dc,v_2}. When user request u is deployed to the data center, for $\forall \bar{v}_1 \in N(u,s), \forall v_1 \in N(dc), \forall p \in P_{\bar{v}_1}^{v_1}, \forall e \in p$, the transmission delay from the source to the data center can be indicated as

$$\begin{cases} t_{u,2}^{tr,1} = \sum_{\bar{v}_1,v_1} q_{u,s}^{\bar{v}_1} \cdot q_{u,s}^{v_1,dc} \cdot (t_{s,\bar{v}_1} + t_{v_1,dc}) + t_{u,2}^{sat,1}, \\ t_{u,2}^{sat,1} = \sum_{\bar{v}_1,v_1} q_{u,s}^{\bar{v}_1} \cdot q_{u,s}^{v_1,dc} \sum_p \sum_e w_{u,p}^{s,2} \cdot t_e. \end{cases} \quad (22)$$

For $\forall v_2 \in N(dc), \forall \bar{v}_2 \in N(u,d), \forall p \in P_{v_2}^{\bar{v}_2}, \forall e \in p$, the transmission delay from the data center to the destination can be indicated as

$$\begin{cases} t_{u,2}^{tr,2} = \sum_{v_2,\bar{v}_2} q_{u,d}^{v_2,dc} \cdot q_{u,d}^{\bar{v}_2} \cdot (t_{dc,v_2} + t_{\bar{v}_2,d}) + t_{u,2}^{sat,2}, \\ t_{u,2}^{sat,2} = \sum_{v_2,\bar{v}_2} q_{u,d}^{v_2,dc} \cdot q_{u,d}^{\bar{v}_2} \sum_p \sum_e w_{u,p}^{|F_u|-1,d} \cdot t_e. \end{cases} \quad (23)$$

Then the transmission delay $t_{u,2}^{tr}$ for user request u, which is deployed to cloud servers, can be indicated as

$$t_{u,2}^{tr} = t_{u,2}^{tr,1} + t_{u,2}^{tr,2}. \quad (24)$$

The total processing delay constraint of user request u can be expressed as

$$t_u^{delay} = t_u^{cmp} + y_u \cdot t_{u,1}^{tr} + (1-y_u) \cdot t_{u,2}^{tr} \le t_u^{max}. \quad (25)$$

In this paper, our optimization goal is to minimize the average satellite network bandwidth cost and the average service end-to-end delay jointly. We indicate the average satellite network bandwidth cost C_{bw} as

$$C_{bw} = \frac{1}{|E|} \sum_e (b_{e,used} + b_{e,1} + b_{e,2}), \forall e \in E. \quad (26)$$

We indicate the average service end-to-end delay C_{user} as

$$C_{user} = \frac{1}{\sum_{u} x_u} \sum_{u} x_u \cdot t_u^{delay}, \forall u \in U. \tag{27}$$

For the VNF placement problem in satellite edge clouds, we consider two optimization sub-problems within the physical constraints of network resources and service requirements. One is to minimize the satellite network bandwidth cost in equation (26). The other is to minimize the service end-to-end delay in equation (27). Then we formulate the two optimization sub-problems as a multi-objective optimization problem, which can be indicated as

$$\begin{aligned} min \quad & C_{total} = \alpha_1 \cdot C_{bw} + \alpha_2 \cdot C_{user}, \\ s.t. \quad & (5) - (8), (13) - (17), (25), \end{aligned} \tag{28}$$

where the cost values for C_{bw} and C_{user} are denoted as α_1 units per Mbps and α_2 units per ms, respectively.

C. Problem Complexity

The generalized assignment problem (GAP) [40] can be reduced to the formulated VNF placement problem. Specifically, we consider satellites and the data center as bins and their resource capacities as bin capacities. Each user request indicates an item and the required resources indicate the item size. All VNFs except the source and the destination for a user request are deployed to only one satellite or the data center. The profit of an item is non-negative and inversely proportional to the deployment cost of a user request, which can be indicated by the network bandwidth cost and the service end-to-end delay. The aim of GAP is to find an optimal placement solution of items with maximum overall profits. The objective of the formulated VNF placement problem can also be converted to maximize the sum of profits for user requests. Therefore, the formulated VNF placement problem is NP-hard due to GAP is NP-hard.

V. Proposed Algorithm

As the VNF placement problem is NP-hard, heuristic algorithms, e.g., Greedy and Simulation Annealing [15], Viterbi [25], and Genetic Algorithm [41], have been widely used to find approximate strategies in a low acceptable time. Therefore, we propose a distributed VNF placement (D-VNFP) algorithm to address the VNF placement problem.

A. Distributed VNF Placement Algorithm

For the proposed D-VNFP algorithm, we consider that satellite v hosts the agent a, which has the characteristics of environment-aware, autonomy, and social behavior [7]. The neighbouring satellites for user requests and the data center serve as their neighbouring agents. The procedure of the proposed D-VNFP algorithm includes two parts of VNF placement strategy and service deployment.

For the VNF placement strategy, the agents can share the network state for managing network resources and making the VNF placement strategies in a self-interested way. To be more specifically, each agent firstly updates the local information about the current satellite network state. When a new user request is to arrive, the information concerning requested service will be randomly sent to one of the source neighbouring agents. The source neighbouring agent can make the VNF placement strategy by the Viterbi algorithm, which minimizes the network bandwidth cost and the service end-to-end delay jointly. It should be noted that the neighbouring agent, which makes the VNF placement strategy, may be not the agent routing the traffic from the user request.

For the service deployment, each user request can be performed by multiple agents in a collaborative way [42]. The VNF placement strategy firstly needs to share with other related agents by the neighbouring agent. Based on the VNF placement strategy, we can place the VNFs to the corresponding agents and build the SFC. Each agent is responsible for running service functions and forwarding data independently, all related agents can collaborate to serve the requested service. When a user request is end, we will break the SFC and release the used resources, e.g., bandwidth, CPU, and memory, to be available for new user requests in the next time slot.

Furthermore, the VNF placement strategies for all user requests are made on different agents concurrently and thus the agents do not know the VNF placement strategies with each other. As a result, the resource requirements of user requests may exceed the available resources of the satellite network, which can result in potential resource conflicts. We use a resource competition approach based on request priority to address the potential resource conflict problem. We consider that the lower is the deployment cost for a user request and the higher is the priority, where user requests can be deployed to satellite edge clouds by the priority-based way in descending order. For an agent, when the required resources of user requests are greater than the available resources, then the user requests will fail to place to satellite edge clouds due to the insufficient resources. In that case, the neighbouring agents will update the local network resource state and perform the Viterbi algorithm for the failed user requests again to find new approximate VNF placement strategies. The iteration will terminate when all user requests are deployed or they fail to deploy without the resource conflict.

Note that the running state of a satellite network for the proposed D-VNFP algorithm needs to be shared among these agents, which can bring a transmission cost for state synchronization. Moreover, each satellite can be responsible for managing network resources and providing computing services autonomously, which can increase the design complexity of each satellite. In addition, due to the computation complexity of the proposed D-VNFP algorithm in large-scale satellite networks and the real-time service requirements of user requests, we consider to divide a large-scale satellite network into several small-scale satellite networks. For each small-scale satellite network, the proposed D-VNFP algorithm can be used to manage network resources and provide computing services for user requests. Furthermore, some user requests can fail to deploy to satellite edge clouds as per the insufficient network resources in each time slot, then we consider that the failed user requests in the current time slot can be deployed again after updating the satellite network state in the next time slot.

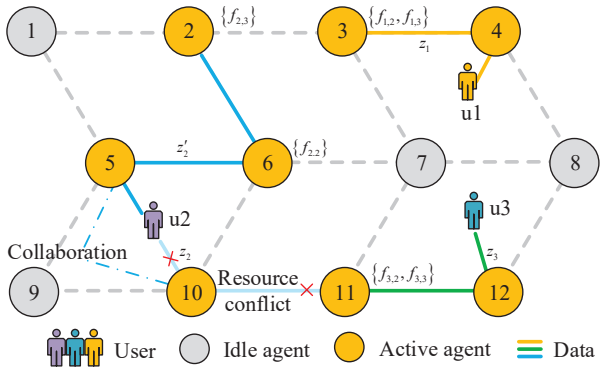


Fig. 3. Procedure of the proposed D-VNFP algorithm.

Fig. 3 illustrates an example of the proposed D-VNFP algorithm. There are 12 satellite agents, i.e., $a_1, a_2, \cdots, a_{12}$, and each agent can communicate with the adjacent agents directly. Three user requests u_1, u_2, and u_3 are in coverage of agent a_4, agents a_5 and a_{10}, and agent a_{12}, respectively. We assume that each user request u can be indicated as $\{f_{u,s}, f_{u,2}, f_{u,3}, f_{u,d}\}$. For user request u_1, the neighbouring agent a_4 firstly updates the local network resource state and obtains the requested service. Then agent a_4 makes the VNF placement strategy z_1, where VNFs $f_{1,2}$ and $f_{1,3}$ are deployed to agent a_3 and the routing path is $s_1 \rightarrow a_4(f_{1,s}) \rightarrow a_3(f_{1,2}, f_{1,3}) \rightarrow a_4(f_{1,d}) \rightarrow d_1$. For the service deployment, agents a_4 and a_3 can share the VNF placement strategy by interacting with each other, and agent a_3 will try to provide the available resources for the two VNFs $f_{1,2}$ and $f_{1,3}$. If agent a_3 can provide sufficient available resources for strategy z_1, then we can build the SFC and provide computing service for user request u_1 in a collaborative way. One of agents a_5 and a_{10} makes the VNF placement strategy z_2 for user request u_2 as $s_2 \rightarrow a_{10}(f_{2,s}) \rightarrow a_{11}(f_{2,2}, f_{2,3}) \rightarrow a_{10}(f_{2,d}) \rightarrow d_2$. Agent a_{12} makes the VNF placement strategy z_3 for user request u_3 as $s_3 \rightarrow a_{12}(f_{3,s}) \rightarrow a_{11}(f_{3,2}, f_{3,3}) \rightarrow a_{12}(f_{3,d}) \rightarrow d_3$. However, there exists the resource conflict for the two VNF placement strategies z_2 and z_3 as the required resources for agent a_{11} are more than the available resources. Agent a_{11} will provide the available resources for user request u_3 with high priority, which can lead to the deployment failure of user request u_2. Then the neighbouring agent of user request u_2 will update the local network resource state and make the new VNF placement strategy z'_2 as $s_2 \rightarrow a_5(f_{2,s}) \rightarrow a_6(f_{2,2}) \rightarrow a_2(f_{2,3}) \rightarrow a_6 \rightarrow a_5(f_{2,d}) \rightarrow d_2$, where we deploy VNF $f_{2,2}$ on agent a_6 and VNF $f_{2,3}$ on agent a_2, respectively.

The proposed D-VNFP algorithm is shown in Algorithm 1. At the beginning, all the agents should share the current network resource state. Then the source neighbouring agents for user requests will make their approximate VNF placement strategies by the Viterbi algorithm in parallel. If the available resources on satellites are insufficient, then we will consider to deploy user requests to cloud servers by the Path Selection algorithm. For the service deployment, we sort all the user requests by deployment cost in ascending order and then try to deploy them one by one from low deployment cost to high deployment cost. When there exists a resource conflict for user request u, we will set the strategy as $z_u = \varnothing$ and put user request u into list U_{next}, which is initialized as $U_{next} = \varnothing$. When there is no resource conflict for user request u, then we can deploy user request u by strategy $z_u^* = z_u$ to satellite edge clouds. After all user requests are deployed, we will reconsider the set of user requests as $U \leftarrow U_{next}$ and perform the proposed D-VNFP algorithm again. When none of user requests needs to be deployed, that is, $U == \varnothing$, the iteration will terminate. If there are insufficient resources for user requests, they will fail to deploy, where the VNF placement strategy for each user request u is considered as $z_u^* = \varnothing$. In that case, the failed user requests can be performed locally or deployed again in the next time slot.

Algorithm 1 Proposed Distributed VNF Placement Algorithm.

Input: User requests U;
Output: $z^* = \left[z_u^* | \forall u \in U\right]$;

Initialize: $U_{next} = \varnothing, \forall u \in U, z_u = z_u^* = \varnothing$;
while $U! = \varnothing$ **do**
  All the agents share the current network resource state;
  for $\forall u \in U$ in parallel **do**
    Search a feasible VNF placement strategy z_u by the *Viterbi* algorithm in satellite edge computing;
    if $z_u == \varnothing$ **then**
      Search a feasible VNF placement strategy z_u by the *Path Selection* algorithm in cloud computing;
    end if
  end for
  Sort user requests by C_{total} in ascending order;
  for $\forall u \in U$ **do**
    if $z_u! = \varnothing$ and there is no resource conflict **then**
      Deploy u by strategy z_u and then $z_u^* \leftarrow z_u$;
    else if $z_u! = \varnothing$ **then**
      Set $z_u = \varnothing$ and then $U_{next} = U_{next} \cup \{u\}$;
    end if
  end for
  $U \leftarrow U_{next}$ and then $U_{next} = \varnothing$;
end while
return z^*;

For the VNF placement problem in satellite edge computing, the Viterbi algorithm [24], [25] is used to minimize the network bandwidth cost and the service end-to-end delay jointly. In order to optimize the network bandwidth cost while guaranteeing the service end-to-end delay, one source neighbouring agent of user request u can be randomly considered as the decision-making node. We can calculate the d shortest paths $P_{s_u}^{d_u}$ by traversing the source and destination neighbouring agents to be the candidate paths and sort them by service delay in ascending order. The source neighbouring agent can traverse the candidate paths one by one and then deploy the VNFs to minimize the network bandwidth cost.

For the Viterbi algorithm, the VNF placement strategy of $f_{u,i}$ is considered as a possible state $\varphi_{u,i}$, where the VNF placement strategy of $f_{u,s}$ is indicated as $\varphi_{u,s}$. All possible states for the same VNF make up the state set Ω in a stage and

Algorithm 2 Viterbi Algorithm.

Input: u;
Output: z_u;

Initialize: $z_u = \varnothing$;
Obtain the candidate paths $P_{s_u}^{d_u}$;
Sort all the paths by end-to-end delay in ascending order;
for each $p \in P_{s_u}^{d_u}$ **do**
  if the delay constraint is not satisfied **then**
    Break;
  end if
  Sort the VNFs except $f_{u,s}$ by topology as Γ_u;
  $\Omega = \{\varphi_{u,s}\}$;
  for each $f_{u,i} \in \Gamma_u$ **do**
    $\Omega' = \varnothing$;
    for each $\varphi \in \Omega$ **do**
      Update the current network resource state by φ;
      List the candidate edge servers $V_{u,i}^{p}$;
      for each $v \in V_{u,i}^{p}$ **do**
        Calculate the required resources;
        if the required resources are satisfied **then**
          Obtain $c_{bw}^{u}, \varphi_{u,i}$, and then $\Omega' = \Omega' \cup \{\varphi_{u,i}\}$;
        end if
      end for
    end for
    Sort Ω' by bandwidth cost in ascending order;
    $\Omega \leftarrow \Omega'[:B]$;
  end for
  if $\Omega! = \varnothing$ **then**
    Obtain the optimal strategy z_u and break;
  end if
end for
return z_u;

Algorithm 3 Path Selection Algorithm.

Input: u;
Output: z_u;

Initialize: $z_{s_u}^{dc} = z_{dc}^{d_u} = \varnothing$;
Obtain the candidate paths $P_{s_u}^{dc}$ and $P_{dc}^{d_u}$;
Sort all the paths by end-to-end delay in ascending order;
for each $p \in P_{s_u}^{dc}$ **do**
  Calculate the bandwidth and service requirements;
  if the required constraints are satisfied **then**
    Obtain the optimal path strategy $z_{s_u}^{dc}$ and break;
  end if
end for
for each $p \in P_{dc}^{d_u}$ **do**
  Calculate the bandwidth and service requirements;
  if the required constraints are satisfied **then**
    Obtain the optimal path strategy $z_{dc}^{d_u}$ and break;
  end if
end for
if $z_{s_u}^{dc}! = \varnothing$ and $z_{dc}^{d_u}! = \varnothing$ **then**
  $z_u = \{z_{s_u}^{dc}, z_{dc}^{d_u}\}$;
end if
return z_u;

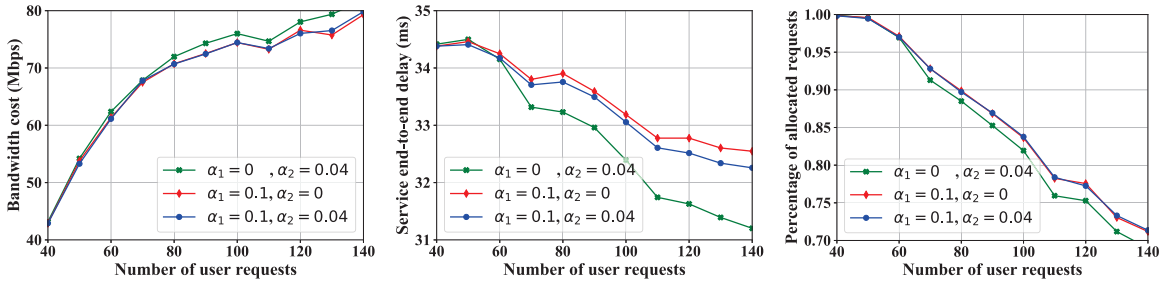
a possible state from Ω is indicated as φ. The edge between different states from adjacent stages represents the network bandwidth cost, and the cumulative bandwidth cost c_{bw}^{u} for the VNF placement strategy is saved in the state node. The number of stages for a user request is equal to the number of the VNFs. After all the VNFs are deployed, the final cumulative network bandwidth cost results can be found in the last destination state nodes. The most likely path with minimum bandwidth cost can be considered as the optimal VNF placement strategy. Note that the Viterbi algorithm is performed under multiple physical constraints as shown in equation (28). Furthermore, to reduce the computation complexity of the Viterbi algorithm, we remain the B optimal possible states for each stage to participate in the VNF placement strategy of the next stage. When the source neighbouring agent finds the first VNF placement strategy z_u with minimum bandwidth cost, the search procedure for user request u will terminate. The procedure of the Viterbi algorithm is shown in Algorithm 2.

For the VNF placement problem in cloud computing, we assume that all the VNFs except the source function and the destination function will be deployed to cloud servers and the processing delay of switches and links in the cloud data center is not considered. The satellite network is just responsible for routing original data from the source to cloud servers and returning the processing results from cloud servers to the destination. In this scenario, our aim is to find the best available paths between the source and cloud servers, and cloud servers and the destination, respectively.

We use the Path Selection algorithm to find the optimal available paths with minimum service end-to-end delay for a user request. The procedure of the Path Selection algorithm is shown in Algorithm 3. For user request u, initially, we consider one of the source neighbouring agents $N(u,s)$ as the access node. Then we calculate the d shortest paths $P_{s_u}^{dc}$ between the source and the data center, and sort them by end-to-end delay in ascending order. Similarly, we can obtain the d shortest paths $P_{dc}^{d_u}$ between the data center and the destination. The source neighbouring agent can traverse the paths $P_{s_u}^{dc}$ from low delay to high delay and calculate the network bandwidth and service requirements. If the bandwidth and service requirements are satisfied, we can consider the current path as the optimal path strategy $z_{s_u}^{dc}$ and break the search procedure. Similarly, we can also obtain the optimal path strategy $z_{dc}^{d_u}$. If $z_{s_u}^{dc}! = \varnothing$ and $z_{dc}^{d_u}! = \varnothing$, the optimal VNF placement strategy can be indicated as $z_u = \{z_{s_u}^{dc}, z_{dc}^{d_u}\}$.

B. Algorithm Complexity

For the proposed D-VNFP algorithm, we assume that M user requests need to be provided computing services in a time slot. In the worst case, only one user request is successfully deployed to satellite edge clouds in each iteration because of resource competition. Thus, the VNF placement procedure for user requests should be run for $\frac{M(M+1)}{2}$ times. The computation complexity of the Viterbi algorithm and the Path selection algorithm can be indicated as $O(dBNF)$ and $O(d)$, where we indicate the maximum number of VNFs for a user request as



(a) Network bandwidth cost. (b) Service end-to-end delay. (c) Percentage of allocated requests.

Fig. 4. Performance comparison for different cost values.

TABLE II
PARAMETER SETTING.

Satellite Network	
Number of satellites	16
Initial inter-satellite link distance	1200 km
Link bandwidths	100 Mbps, 1 Gbps
Edge server	96 vCPUs, 112 GB memory
User Requests	
Number of VNFs	Truncated power-law distribution
Required resources per VNF	[2,4] vCPUs, [4,8] GB memory
Required bandwidths per edge	[10,20] Mbps
Execution time per VNF	[5,10] ms

F. Therefore, we can obtain the computation complexity of the proposed D-VNFP algorithm to be $O(M^2dBNF)$.

VI. PERFORMANCE EVALUATION

In this section, we conduct the experiments to evaluate the performance of the proposed D-VNFP algorithm in satellite edge clouds. The parameters for a satellite network are generated by AGI Systems Tool Kit (STK) [43]. The third packages *Skyfield* [44] and *Networkx* [45] are used to build the satellite network, which considers the mobility of satellites. All experiments are implemented by Python and run on a general computing platform, which includes I7-4790K CPU, 16 GB memory, and Windows 10.

A. Simulation Setup

A satellite network with $N = 16$ LEO satellites is used in our simulation experiments, where there are 4 orbital planes and each orbital plane has 4 LEO satellites. For each satellite, the altitude is 780 km, the available resources include 96 vCPUs and 112 GB memory. The initial distance of inter-satellite links is about 1200 km. The bandwidth capacity of each inter-satellite link is 100 Mbps. The cloud data center is in the coverage of *Sat10*, where there are sufficient resources for CPU and memory. The satellite-ground link bandwidth capacity is 1 Gbps.

For a user request, we assume the number of the VNFs follows a truncated power-law distribution, where the exponent is 2, the minimum value is 2, and the maximum value is 7 [46]. The resource requirements for each VNF are randomly generated from $[2, 4]$ vCPUs and $[4, 8]$ GB memory. The execution time for each VNF is $[5, 10]$ ms. The bandwidth requirements for each edge between the adjacent VNFs are $[10, 20]$ Mbps. The source and the destination are randomly generated in the coverage of satellites. The maximum acceptable delay can guarantee that user requests are offloaded to the cloud data center for further processing. The main simulation parameters are summarized in Table II. According to the evaluation results in [24], we set the number of the shortest paths between two satellites as $d = 8$ and the search width of the Viterbi algorithm as $B = 4$. Moreover, we suppose the cost values in equation (28) as $\alpha_1 = 0.1$ units per Mbps and $\alpha_2 = 0.04$ units per ms, respectively. Besides, we just evaluate the performance of the proposped D-VNFP algorithm, where the message synchronization delay in a satellite network is not considered.

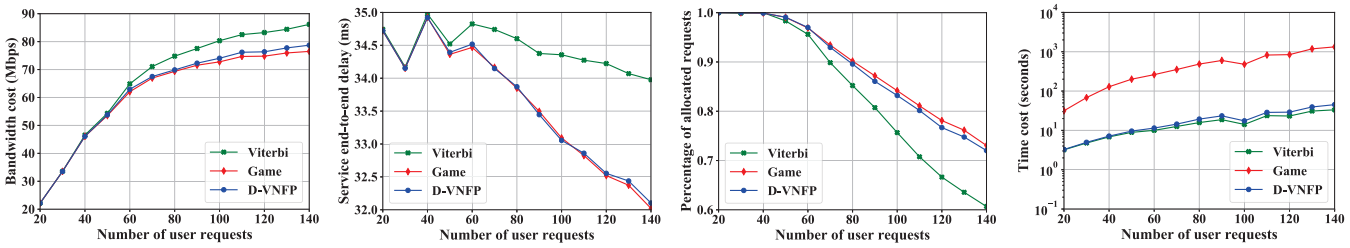
Moreover, two existing algorithms of Game theory [23] and Viterbi [25] are considered as the baseline algorithms to conduct the experiments.

Game theory: As a non-cooperative game, potential game [23] is widely used to solve the problem of resource allocation in a decentralized way. Each player makes the VNF placement strategy in self-interested manner, i.e., only one player with maximum user payoff can win the chance to update the strategy in each iteration. A Nash equilibrium strategy profile will be found in a finite number of iterations.

Viterbi: Viterbi [25] is used to tackle the VNF placement problem, which is viewed as part of the proposed D-VNFP algorithm. In each time slot, we can provide computing services for user requests one by one, where the Viterbi algorithm and the Path Selection algorithm are used for each user request to seek an approximate VNF placement strategy.

B. Performance Comparison in Static Environment

To evaluate the performance of the proposed D-VNFP algorithm for different cost values of α_1 and α_2, the experiments with $\alpha_1 = 0, \alpha_2 = 0.04$, $\alpha_1 = 0.1, \alpha_2 = 0$, and $\alpha_1 = 0.1, \alpha_2 = 0.04$ are conducted by the proposed D-VNFP algorithm, respectively. The number of user requests is denoted as $M = \{40, 50, \cdots, 140\}$, each experiment runs for 30 times and we obtain the average results, which are shown in Fig. 4. For $\alpha_1 = 0, \alpha_2 = 0.04$, the objective in equation (28) is just to minimize the service end-to-end delay. In that case, the average service end-to-end delay can be reduced but the average network bandwidth cost will be increased, which can decrease the number of allocated user requests. For $\alpha_1 = 0.1, \alpha_2 = 0$, the objective in equation (28) is just to



(a) Network bandwidth cost. (b) Service end-to-end delay. (c) Percentage of allocated requests. (d) Time cost.

Fig. 5. Performance comparison for D-VNFP, Viterbi, and Game Theory.

minimize the network bandwidth cost. In that case, the average network bandwidth cost can be reduced but the average service end-to-end delay will be increased, which can also increase the number of allocated user requests as the available network resources increase. For $\alpha_1 = 0.1, \alpha_2 = 0.04$, the objective in equation (28) can minimize the network bandwidth cost and the service end-to-end delay jointly, as shown in Fig. 4. Therefore, we can adjust the preference between network bandwidth cost and service end-to-end delay by changing the cost values of α_1 and α_2.

We also conduct the following experiments to evaluate the performance of the proposed D-VNFP algorithm compared with Game theory and Viterbi. The number of user requests for the experiments is set as $M = \{20, 30, \cdots, 140\}$. Each experiment runs for 30 times and we obtain the average results.

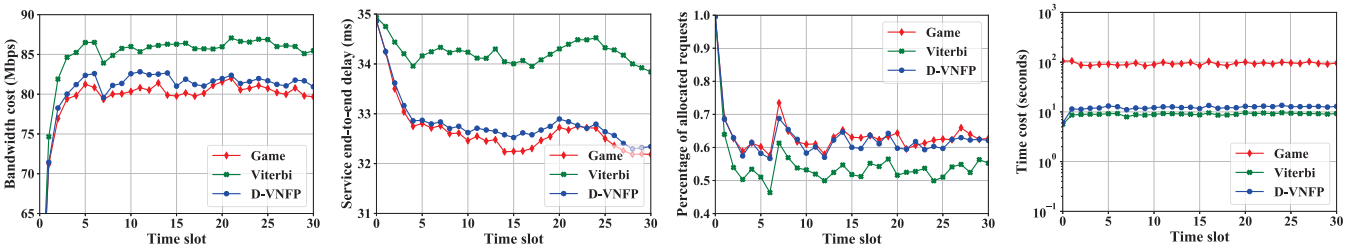
The simulation results for different user requests are shown in Fig. 5. Fig. 5(a) illustrates the average network bandwidth costs for different user requests. When the number of user requests is small, e.g., $M = \{20, 30\}$, there are sufficient resources for deploying user requests to satellite edge clouds by D-VNFP, Viterbi, and Game theory. Therefore, the performance of the three VNF placement algorithms is relatively close. For an example of $M = 30$, the network bandwidth costs for Viterbi, D-VNFP, and Game theory are 33.55 Mbps, 33.51 Mbps, and 33.51 Mbps, respectively. As the number of user requests increases, the available resources are gradually decreasing, which can result in the deployment failure of user requests due to the resource limitation.

For the Viterbi algorithm, user requests are deployed to satellite edge clouds one by one in a greedy way, which can not guarantee that user requests with low deployment costs are preferentially deployed. Therefore, the performance of the Viterbi algorithm is the worst. For the proposed D-VNFP algorithm, all the agents make the VNF placement strategies for user requests in parallel. For each iteration, all user requests will be deployed to satellite edge clouds by a priority-based approach. Thus, the proposed D-VNFP algorithm will outperform the Viterbi algorithm. For Game theory, the players make the VNF placement strategies in a self-interested way. Only one user request with maximum user payoff, which is also considered as minimum deployment cost, can win the chance to update the strategy for each iteration until a Nash equilibrium appears. That is, user requests with minimum deployment costs will be deployed to satellite edge clouds. However, it can not guarantee for the proposed D-VNFP algorithm that all user requests with minimum deployment costs are preferentially deployed as a priority-based approach is used to solve the resource conflict problem. In that case, Game theory can perform better than the Viterbi algorithm and the proposed D-VNFP algorithm. Note that the proposed D-VNFP algorithm performs close to the Game theory approach. On average, the network bandwidth cost obtained by the proposed D-VNFP algorithm can reduce by 5.85% for Viterbi and just increase by 1.41% for Game theory, respectively.

Fig. 5(b) describes the service end-to-end delay for different user requests. Similar to the results in Fig. 5(a), we can observe that the service end-to-end delay of Viterbi is the worst, followed by the D-VNFP algorithm and finally Game theory, where the service end-to-end delay of the proposed D-VNFP algorithm is close to that of Game theory. For all the cases, the average service end-to-end delay of the proposed D-VNFP algorithm decreases by 2.38% for Viterbi. However, the performance difference between the proposed D-VNFP algorithm and Game theory is just 0.04%.

Fig. 5(c) shows the percentages of allocated user requests for different user requests. As the number of user requests is small, available resources are sufficient for user requests. Thus, all user requests can be deployed to satellite edge clouds. When the number of user requests increases, the available resources will gradually reduce. In that case, if the resource requirements of user requests exceed the available resources of satellite edge clouds, user requests will fail to deploy, which can lead to the decrease of allocated user requests. Generally, the better is the performance of the VNF placement algorithm, the less are the deployment costs for user requests, and then the more is the number of allocated user requests. Therefore, we can observe from Fig. 5(c) that the percentages of allocated user requests for Viterbi, D-VNFP, and Game theory become better in turn. For all the cases, the average percentage of allocated user requests of the proposed D-VNFP algorithm increases by 5.93% for Viterbi and just decreases by 0.67% for Game theory.

Fig. 5(d) shows the time costs for different user requests. We can observe that the time cost of the proposed D-VNFP algorithm is close to that of Viterbi but obviously less than that of Game theory. That is due to the fact that Viterbi can deploy user requests one by one in a greedy way, however, D-VNFP and Game theory deploy user requests in an iterative way, respectively. For the proposed D-VNFP algorithm, we make the VNF placement strategies for user requests in parallel and deploy the user requests to satellite edge clouds as much as possible in each iteration. For Game theory, just one user



(a) Network bandwidth cost. (b) Service end-to-end delay. (c) Percentage of allocated requests. (d) Time cost.

Fig. 6. Simulation results for $\lambda_P = 40$ in dynamic environment.

TABLE III
SIMULATION RESULTS FOR DIFFERENT USER REQUESTS IN DYNAMIC ENVIRONMENT.

λ_P	Network Bandwidth Cost (Mbps)			Service End-to-End Delay (ms)			Percentage of Allocated Requests			Time Cost (seconds)		
	Viterbi	D-VNFP	Game	Viterbi	D-VNFP	Game	Viterbi	D-VNFP	Game	Viterbi	D-VNFP	Game
10	44.28	43.9	43.83	34.93	34.86	34.86	0.99	0.99	0.99	2	2	10
15	61.79	59.91	59.69	34.87	34.52	34.51	0.93	0.94	0.94	3	4	22
20	71.32	68.02	67.57	34.62	34.01	34	0.83	0.86	0.87	4	6	36
25	76.72	72.98	71.94	34.47	33.61	33.56	0.74	0.79	0.8	6	7	50
30	80.04	75.79	74.65	34.21	33.17	33.07	0.67	0.73	0.74	7	10	71
35	82.93	78.57	77.78	34.11	32.81	32.79	0.6	0.67	0.68	8	11	81
40	84.65	80.45	79.18	34.09	32.64	32.51	0.55	0.63	0.64	9	13	97
45	86.19	81.67	80.5	33.94	32.26	32.16	0.51	0.59	0.6	10	14	114
50	87.15	82.37	81.18	33.78	32.06	31.98	0.47	0.56	0.57	11	16	128

request can win the chance to be deployed in each iteration, which can result in high computation complexity for finding a Nash equilibrium strategy profile. For all the cases, the average time cost of the D-VNFP algorithm increases by 22.79% for Viterbi but decreases by 96.26% for Game theory.

C. VNF Placement in Dynamic Environment

To evaluate the on-line performance of the proposed D-VNFP algorithm for deploying user requests, we conduct the following experiments in dynamic environment. We assume that the number of user requests to appear in a time slot follows the Poisson distribution with $\lambda_P = \{10, 15, \cdots, 50\}$ and the running time of each user request follows the Exponent distribution with $\lambda_E = 3$. The number of time slots for each experiment is 50. We run each experiment for 30 times and obtain the average results.

For an example of $\lambda_P = 40$, the simulation results for different time slots are shown in Fig. 6. Fig. 6(a) depicts the network bandwidth costs for deploying user requests over different time slots. We can observe that the proposed D-VNFP algorithm for all time slots performs better than Viterbi but approximates to Game theory. The average network bandwidth cost of the proposed D-VNFP algorithm decreases by 4.96% for Viterbi and just increases by 1.59% for Game theory. Fig. 6(b) and Fig. 6(c) describe the service end-to-end delay and the percentages of allocated user requests over different time slots, respectively. Overall, the performance of Viterbi is the worst, followed by D-VNFP, and finally Game theory. The average service end-to-end delay obtained by the proposed D-VNFP algorithm reduces by 4.24% for Viterbi and just increases by 0.42% for Game theory. The average percentage of allocated user requests obtained by the proposed D-VNFP algorithm improves by 14.11% for Viterbi and just reduces by 1.97% for Game theory. In addition, the time costs for deploying user requests over different time slots are illustrated in Fig. 6(d). We can find that the proposed D-VNFP algorithm increases the time cost by 37.81% for Viterbi but reduces the time cost by 86.95% for Game theory.

In order to further demonstrate the effectiveness of the proposed D-VNFP algorithm, we provide the average simulation results for different user requests in dynamic environment, as shown in Table III. We can also observe from Table III that the proposed D-VNFP algorithm outperforms Viterbi but approximates to Game theory in terms of network bandwidth cost, service end-to-end delay, and percentage of allocated user requests. Specifically, for all the cases, the average network bandwidth cost obtained by the proposed D-VNFP algorithm reduces by 4.65% for Viterbi and just increases by 1.15% for Game theory. The average service end-to-end delay obtained by the proposed D-VNFP algorithm saves by 2.93% for Viterbi and just increases by 0.16% for Game theory. The average percentage of allocated user requests obtained by the proposed D-VNFP algorithm increases by 7.48% for Viterbi and just decreases by 1.20% for Game theory. Note that the time cost of the proposed D-VNFP algorithm is close to that of Viterbi but obviously better than that of Game theory. On average, the proposed D-VNFP algorithm increases the time cost by 33.87% for Viterbi and can reduce the time cost by 86.49% for Game theory.

According to the experiments, we can demonstrate the effectiveness of the proposed D-VNFP algorithm for solving the VNF placement problem in satellite edge clouds. For Viterbi, the proposed D-VNFP algorithm can improve the performance of solving the VNF placement problem in network bandwidth cost, service end-to-end delay, and percentage of allocated user requests while keeping low time cost. For Game theory, the performance is close to that of the proposed D-

VNFP algorithm in network bandwidth cost, service end-to-end delay, and percentage of allocated user requests. However, the time cost of Game theory is obviously greater than that of the proposed D-VNFP algorithm. That is, the proposed D-VNFP algorithm has a tradeoff between strategy quality and computation complexity when compared with Game theory.

VII. Conclusion

In this paper, we investigated the VNF placement problem in satellite edge clouds with minimizing the network bandwidth cost and the service end-to-end delay jointly. We formulated the VNF placement problem as an integer nonlinear programming problem, which is viewed as NP-hard. Then we proposed a distributed VNF placement algorithm to address the problem. For each user request, we used the Viterbi algorithm and the Path Selection algorithm to find an approximate VNF placement strategy, respectively. The VNF placement strategies for user requests can be made in a distributed and parallel manner. We used a priority-based approach to solve the possible resource conflict.

To evaluate the performance of the proposed D-VNFP algorithm, we conducted the experiments in satellite edge cloud scenarios and compared the performance with two baseline algorithms of Viterbi and Game theory. The simulation results show the validity of the proposed D-VNFP algorithm for deploying user requests to satellite edge clouds. Specifically, the proposed D-VNFP algorithm performs better than Viterbi and approximates to Game theory. However, the running time of the proposed D-VNFP algorithm is less than that of Game theory and close to that of Viterbi.

References

[1] M. De Sanctis *et al.*, "Satellite communications supporting Internet of Remote Things," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 113–123, 2016.

[2] L. Zhen *et al.*, "Energy-efficient random access for LEO satellite-assisted 6G Internet of Remote Things," *IEEE Internet Things J.*, 2020, doi: 10.1109/JIOT.2020.3030856.

[3] R. Xie *et al.*, "Satellite-terrestrial integrated edge computing networks: Architecture, challenges, and open issues," *IEEE Network*, vol. 34, no. 3, pp. 224–231, 2020.

[4] F. Xu *et al.*, "Deep reinforcement learning based joint edge resource management in maritime network," *China Commun.*, vol. 17, no. 5, pp. 211–222, 2020.

[5] G. Cui *et al.*, "Joint offloading and resource allocation for satellite assisted vehicle-to-vehicle communication," *IEEE Syst. J.*, 2020, doi: 10.1109/JSYST.2020.3017710.

[6] N. Saeed *et al.*, "Cubesat communications: Recent advances and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1839–1862, 2020.

[7] X. Gao *et al.*, "Hierarchical multi-agent optimization for resource allocation in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 692–707, 2021.

[8] M. Shabbir *et al.*, "Enhancing security of health information using modular encryption standard in mobile cloud computing," *IEEE Access*, vol. 9, pp. 8820–8834, 2021.

[9] C. Qiu *et al.*, "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, 2019.

[10] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, 2019.

[11] L. Ruan *et al.*, "Priority-based residential energy management with collaborative edge and cloud computing," *IEEE Trans. Ind. Inform.*, vol. 16, no. 3, pp. 1848–1857, 2020.

[12] J. Pan *et al.*, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, 2018.

[13] J. Ren *et al.*, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, 2019.

[14] A. Fischer *et al.*, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 2013.

[15] J. Liu *et al.*, "Improve service chaining performance with optimized middlebox placement," *IEEE Trans. Netw. Serv. Manag.*, vol. 10, no. 4, pp. 560–573, 2017.

[16] J. Li *et al.*, "Reliability-aware network service provisioning in mobile edge-cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1545–1558, 2020.

[17] Y. Shih *et al.*, "An NFV-based service framework for IoT applications in edge computing environments," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 4, pp. 1419–1434, 2019.

[18] M. Huang *et al.*, "Reliability-aware virtualized network function services provisioning in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2699–2713, 2020.

[19] S. Yang *et al.*, "Delay-aware virtual network function placement and routing in edge clouds," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 445–459, 2021.

[20] A. Leivadeas *et al.*, "VNF placement optimization at the edge and cloud," *Future Internet*, vol. 11, no. 3:69, 2019.

[21] Z. Zhang *et al.*, "Satellite mobile edge computing: Improving QoS of high-speed satellite-terrestrial networks using edge computing techniques," *IEEE Netw.*, vol. 33, no. 1, pp. 70–76, 2019.

[22] Z. Jia *et al.*, "Joint optimization of VNF deployment and routing in software defined satellite networks," in *Proc. IEEE Veh. Technol. Conf.*, Chicago, USA, Aug. 2018, pp. 1–5.

[23] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, 2020.

[24] X. Gao *et al.*, "Virtual network function placement in satellite edge computing with a potential game approach," *IEEE Trans. Netw. Serv. Manag.*, 2022, doi: 10.1109/TNSM.2022.3141165.

[25] F. Bari *et al.*, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 4, pp. 725–739, 2016.

[26] S. T. Arzo *et al.*, "Study of virtual network function placement in 5G cloud radio access network," *IEEE Trans. Netw. Serv. Manag.*, 2020, doi: 10.1109/TNSM.2020.3020390.

[27] B. Feng *et al.*, "Enabling efficient service function chains at terrestrial-satellite hybrid cloud networks," *IEEE Netw.*, vol. 33, no. 6, pp. 94–99, 2019.

[28] R. Ferrus *et al.*, "On the virtualization and dynamic orchestration of satellite communication services," in *Proc. IEEE Veh. Technol. Conf.*, Montreal, Canada, Sep. 2016, pp. 1–5.

[29] Z. Jia *et al.*, "VNF-based service provision in software defined LEO satellite networks," *IEEE Trans. Wireless Commun.*, 2021, doi: 10.1109/TWC.2021.3072155.

[30] Y. Cai *et al.*, "An approach to deploy service function chains in satellite networks," in *IEEE/IFIP Netw. Oper. Manag. Symp.*, Taipei, Taiwan, Apr. 2018, pp. 1–7.

[31] G. Li *et al.*, "Horizontal-based orchestration for multi-domain SFC in SDN/NFV-enabled satellite/terrestrial networks," *China Commun.*, vol. 15, no. 5, pp. 77–91, 2018.

[32] S. Song *et al.*, "Clustered virtualized network functions resource allocation based on context-aware grouping in 5G edge networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 5, pp. 1072–1083, 2020.

[33] Y. Ma *et al.*, "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks," *IEEE Trans. Mobile Comput.*, 2020, doi: 10.1109/TMC.2020.3006507.

[34] Z. Xu *et al.*, "NFV-Enabled IoT service provisioning in mobile edge clouds," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 1892–1906, 2021.

[35] F. Tashtarian *et al.*, "Distributed VNF scaling in large-scale datacenters: An ADMM-based approach," in *Proc. IEEE Int. Conf. Commun. Technol.*, Chengdu, China, Oct. 2017, pp. 471–480.

[36] Y. Yu *et al.*, "Network function virtualization resource allocation based on joint benders decomposition and ADMM," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1706–1718, 2020.

[37] S. D'Oro *et al.*, "Exploiting congestion games to achieve distributed service chaining in NFV networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 2, pp. 407–420, 2017.

[38] S. Bian *et al.*, "Service chain composition with failures in NFV systems: A game-theoretic perspective," in *Proc. IEEE Int. Conf. Commun.*, Shanghai, China, May 2019, pp. 1–6.
[39] I. Leyva-Mayorga *et al.*, "Inter-plane inter-satellite connectivity in dense LEO constellations," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3430–3443, 2021.
[40] C. Chekuri *et al.*, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM J. Comput.*, vol. 35, no. 3, pp. 713–728, 2006.
[41] W. Rankothge *et al.*, "Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 2, pp. 343–356, 2017.
[42] X. Zheng *et al.*, "A multi-agent optimization algorithm for resource constrained project scheduling problem," *Expert Syst. Appl.*, vol. 42, no. 15-16, pp. 6039–6049, 2015.
[43] Systems tool kit (STK). [Online]. Available: https://www.agi.com/products/stk
[44] B. Rhodes, "Skyfield: High precision research-grade positions for planets and earth satellites generator," *ascl*, vol. ascl: 1907.024, Jul. 2019.
[45] A. Hagberg. Networkx: Network analysis in python. [Online]. Available: https://github.com/networkx
[46] W. Rankothge *et al.*, "Data modelling for the evaluation of virtualized network functions resource allocation algorithms," *ArXiv*, vol. abs/1702.00369, Feb. 2017.

Xiangqiang Gao received the B.Sc. degree in the School of Electronic Engineering from Xidian University and the M.Sc. degree from Xi'an Microelectronics Technology Institute, Xi'an, China, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering, Beihang University, Beijing, China. His research interests include rateless codes, software defined network and network function virtualization.

Rongke Liu received the B.S. and Ph.D. degrees from Beihang University in 1996 and 2002, respectively. He was a Visiting Professor with the Florida Institution of Technology, USA, in 2006; The University of Tokyo, Japan, in 2015; and the University of Edinburgh, U.K., in 2018, respectively. He is currently a Full Professor with the School of Electronic and Information Engineering, Beihang University. He received the support of the New Century Excellent Talents Program from the Minister of Education, China. He has attended many special programs, such as China Terrestrial Digital Broadcast Standard. He has published over 100 papers in international conferences and journals. He has been granted over 20 patents. His research interest covers wireless communication and space information network.

Aryan Kaushik is currently an Assistant Professor (Lecturer) at the University of Sussex, UK, with the School of Engineering and Informatics, from 2021. He has been a Research Fellow at the University College London, UK, with the Department of Electronic and Electrical Engineering, from 2020-21. He completed his PhD degree in Communications Engineering from The University of Edinburgh, UK, in 2019. He received his MSc degree in Telecommunications from The Hong Kong University of Science and Technology, Hong Kong, in 2015. He has held visiting research appointments at the Athena Research and Innovation Center, Greece, in 2021, the Imperial College London, UK, from 2019-20, the University of Luxembourg, Luxembourg, in 2018, and the Beihang University, China, from 2017-19. He has been a TPC Member for IEEE international conferences such as IEEE ICC 2021 and ICC 2022, and a Conference Champion for IEEE PIMRC 2020. His research interests include 5G6G wireless communications, signal processing, integrated communications and radar sensing, millimeter wave, multi-antenna communications and satellite communication networks.

Hangyu Zhang received the B.S. degree in the School of Communication Engineering from Jilin University in 2019. She has been working on the Ph.D. degree in the School of Electronic and Information Engineering, Beihang University, Beijing, China. Her research interest includes the development of machine learning in Satellite Internet.