QoS-Aware VNF Placement and Service Chaining for IoT Applications in Multi-Tier Mobile Edge Networks

ZICHUAN XU and ZHIHENG ZHANG, Dalian University of Technology
WEIFA LIANG, Australian National University
QIUFEN XIA, Dalian University of Technology
OMER RANA, Cardiff University
GUOWEI WU, Dalian University of Technology

Mobile edge computing and network function virtualization (NFV) paradigms enable new flexibility and possibilities of the deployment of extreme low-latency services for Internet-of-Things (IoT) applications within the proximity of their users. However, this poses great challenges to find optimal placements of virtualized network functions (VNFs) for data processing requests of IoT applications in a multi-tier cloud network, which consists of many small- or medium-scale servers, clusters, or cloudlets deployed within the proximity of IoT nodes and a few large-scale remote data centers with abundant computing and storage resources. In particular, it is challenging to jointly consider VNF instance placement and routing traffic path planning for user requests, as they are not only delay sensitive but also resource hungry.

In this article, we consider admissions of NFV-enabled requests of IoT applications in a multi-tier cloud network, where users request network services by issuing service requests with service chain requirements, and the service chain enforces the data traffic of the request to pass through the VNFs in the chain one by one until it reaches its destination. To this end, we first formulate the throughput maximization problem with the aim to maximize the system throughput. We then propose an integer linear program solution if the problem size is small; otherwise, we devise an efficient heuristic that jointly takes into account VNF placements to both cloudlets and data centers and routing path finding for each request. For a special case of the problem with a set of service chains, we propose an approximation algorithm with a provable approximation ratio. Next, we also devise efficient learning-based heuristics for VNF provisioning for IoT applications by incorporating the mobility and energy conservation features of IoT devices. We finally evaluate the performance of the proposed algorithms by simulations. The simulation results show that the performance of the proposed algorithms is promising.

The work of Z. Xu, Q. Xia, and G. Wu was partially supported by the National Natural Science Foundation of China (grant nos. 61802048, 61802047, 61772113, and 61872053), the fundamental research funds for the central universities in China (grant nos. DUT17RC(3)061, DUT17RC(3)070, DUT19RC(4)035, and DUT19GJ204), DUT-RU Co-Research Center of Advanced ICT for Active Life, and the "Xinghai Scholar Program" in Dalian University of Technology, China. The work of W. Liang was supported by the Australian Research Council Discovery Project (grant no. DP200101985).

Authors' addresses: Z. Xu, Z. Zhang, and G. Wu, School of Software, Dalian University of Technology, Dalian, Liaoning, 116621, China; emails: z.xu@dlut.edu.cn, zhangzhiheng@mail.dlut.edu.cn, wgwdut@dlut.edu.cn; W. Liang, Research School of Computer Science, Australian National University, Canberra, ACT 2601, Australia; email: wliang@cs.anu.edu.au; Q. Xia, International School of Information Science and Engineering, Dalian University of Technology, Dalian, Liaoning, 116621, China; email: qiufenxia@dlut.edu.cn; O. Rana, School of Computer Science and Informatics, Cardiff University, Cardiff, CF24 3AA, United Kingdom; email: RanaOF@cardiff.ac.uk.

1550-4859/2020/05-ART23 $15.00
https://doi.org/10.1145/3387705

CCS Concepts: • **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

Additional Key Words and Phrases: Internet of Things, mobile edge clouds, network function virtualization, quality of services, approximation algorithms

ACM Reference format:
Zichuan Xu, Zhiheng Zhang, Weifa Liang, Qiufen Xia, Omer Rana, and Guowei Wu. 2020. QoS-Aware VNF Placement and Service Chaining for IoT Applications in Multi-Tier Mobile Edge Networks. *ACM Trans. Sen. Netw.* 16, 3, Article 23 (May 2020), 27 pages.
https://doi.org/10.1145/3387705

1 INTRODUCTION

With the advancement of wireless access technologies, such as Wi-Fi and 5G, various smart Internet-of-Things (IoT) applications, such as smart home, smart city, smart grid, connected health, and connected cars, have been receiving much attention. These applications usually are driven by AI and need to process large volumes of data in real time to discover valuable patterns hidden in the IoT data. Orthogonal to 5G technology, mobile edge clouds have emerged as an enabling technique to support real-time IoT applications by bringing remote cloud services to nearby IoT applications. However, considering that mobile edge clouds are usually deployed at fixed locations within the proximity of users, such as industrial plants, base stations, shopping malls, and airports, they have limited computing and storage resources due to space limitations of the locations. This unfortunately limits the capability of mobile edge clouds to meet ever-growing resource demands of various network services. *Multi-tier cloud networks* that consist of both mobile edge clouds and remote clouds are ideal platforms to meet not only the extreme low-latency requirements but also resource demands of IoT applications. Specifically, low-latency IoT applications can be moved to the mobile edge cloud while placing resource-hungry services in remote clouds or data centers.

However, IoT applications in multi-tier cloud networks need various network services with different sequences of network functions, such as firewalls and intrusion prevention/detection systems, to ensure the secure, flexible, and low-cost processing and transmission of their traffic. Network function virtualization (NFV) is a promising technology that offers new flexibility in hosting IoT services in any virtualized network node, such as access points (APs), routers, and remote data centers. Specifically, through decoupling network functions from the underlying hardware and implementing them as software running in virtual machines (VMs), NFV reduces the capital expenses (CapEx) and operational expenses (OpEx) of network service providers. Along with the multi-tier cloud network, NFV also creates a flexibility of instantiating IoT services according to their nature (i.e., delay sensitive or resource hungry). For example, virtualized network functions (VNFs) that require a huge amount of resources to process traffic and have a small amount of output traffic can be placed to remote data centers, whereas the VNFs that need a large amount of input data from end users can be placed into the edge cloud to shorten the transmission delay. In this article, we will investigate a fundamental problem of VNF provisioning for IoT applications in a multi-tier cloud network.

Although the NFV-enabled multi-tier cloud network brings the mentioned benefits and promised flexibilities for many real-time IoT applications, it also poses the following challenges. First, IoT applications need resources from both edge clouds and remote data centers to meet their delay and resource requirements. It poses a great challenge to provision VNF service for an IoT application by jointly considering the local edge cloud and remote clouds, such as how to jointly decide which locations (cloudlets or data centers) for VNFs and chain the placed VNFs in a multi-tier cloud network. The VNF locations and routing paths among the locations should be jointly

determined to guarantee that as many as user requests with service chain requirements can be admitted while their accumulative implementation cost is minimized. Second, the implementation of a service chain of a request makes use of not only existing VNFs in data centers but also the newly created VNF instances in cloudlets in the edge cloud network, and thus allowing IoT applications to share existing idle VNFs is challenging. Third, IoT applications usually require extra value-added network functions such as parental controls in smart home applications, caching, and data preprocessing in wireless sensor network applications. How to jointly place such network functions and conventional network functions is challenging. Fourth, IoT applications have their distinct features, such as mobility and energy awareness [29, 32]. This makes the provisioning of VNFs in the multi-tier cloud network extremely challenging, because placed VNFs need to be adaptive to the mobility and energy status of IoT devices so that the performance of the IoT applications can be guaranteed. Fifth, how to guarantee the performance of network services is challenging, by meeting end-to-end delay requirements of user requests of IoT applications, here the delay is composed of the processing delay of VNFs in a service chain and the transmission delay in the routing paths.

There are several studies on VNF placement and service chaining in software-defined or NFV-enabled networks [5–7, 14–17, 19, 22–24, 26, 35, 36, 38]. Some of them have assumed that all VNFs in a service chain are consolidated into a single location. Most of them did not consider the NFV provisioning in multi-tier cloud networks and thus ignored the resource orchestration of both edge and remote clouds for IoT applications. Although there are a few studies on provisioning VNFs for IoT applications [3, 9, 13, 27], they either have not considered the chaining of VNFs or have ignored the distinct features (i.e., mobility) of IoT applications. Unlike the aforementioned studies, we jointly consider the service chaining and VNF placement, mobility, and energy awareness of IoT applications by exploring a collaborative resource orchestration of both edge and remote clouds.

In this article, we take into account the benefits and flexibility afforded by the NFV technique for IoT applications along with the mentioned challenges. We study the throughput maximization problem in a multi-tier cloud network, with the aim to improve resource utilization of the multi-tier cloud network, by admitting as many NFV-enabled requests as possible. To the best of our knowledge, we are the first to study QoS-aware service chaining for IoT applications in a multi-tier cloud network consisting of cloudlets and remote data centers, by leveraging a non-trivial trade-off between request implementation costs and end-to-end delays. We jointly place VNFs into cloudlets and data centers and chain the placed VNFs together through finding routing paths among the locations via a smart auxiliary graph construction. We also consider dynamic VNF placements to allow the obtained solution to be adaptive to both mobility and energy statuses of IoT devices.

The main contributions of this work are as follows:

- We study the service chaining problem for IoT applications in a multi-tier cloud network with the aim to maximize the system throughput while minimizing the implementation cost of admitted requests, subject to capacity constraints on cloudlets.
- We formulate an integer linear program (ILP) for the throughput maximization problem.
- We devise an efficient heuristic for the problem through reducing it to an unsplittable min-cost multi-commodity flow problem, due to the poor scalability of the ILP solution.
- We propose an approximation algorithm with a provable approximation ratio for a special case of the problem without end-to-end delay requirements.
- We devise efficient learning-based heuristics to deal with the mobility and energy statuses of IoT devices.
- We evaluate the performance of the proposed algorithms based on simulations, and the performance of our simulations is promising.

The rest of the article is organized as follows. Section 2 surveys the state of the art on this topic and details the difference between this work and previous studies of task offloading. Section 3 introduces the system model, notations, and problem definitions. Section 4 provides an ILP formulation and then devises a heuristic algorithm for the problem. For a special case of service chains, Section 5 proposes an approximation algorithm with a provable approximation ratio for the problem. Section 6 devises an efficient and learning-based heuristic to deal with the mobility and energy statuses of IoT devices. Section 7 presents some experimental results on the performance of the proposed algorithms, and Section 8 concludes the article.

2 RELATED WORK

Recently, NFV-enabled request admissions and mobile edge computing have attracted much attention in the literature [5–7, 14–17, 19, 22–24, 26, 35, 36] due to the promises of the NFV and mobile edge cloud techniques and the new challenges they have brought. Most NFV studies have focused on hybrid networks with both software-defined network (SDN)-enabled and conventional switches, SDNs, data center networks, or distributed clouds with both hardware and software network functions [26], online algorithm design for dynamic networks [22, 23], and delay-awareness [21], by proposing exact solutions [22], approximation solutions [7], heuristics [26], or online algorithms [16]. However, most studies on mobile edge clouds have focused on task offloading [18] and load balancing among different cloudlets [19]. The service chaining of VNFs in mobile edge networks is usually ignored. For example, Song et al. [30] investigated the task assignment problem in a mobile edge network with node, link, and security constraints. Chen and Wu [5] investigated a series of innovative algorithms for NFV middlebox placement by considering the balance of the setup and bandwidth consumption costs.

There are several studies on the provisioning of NFV-enabled network services in mobile edge clouds [18, 28, 40, 41]. For example, Yang et al. [40, 41] studied the problem of placing VNF instances among NFV-enabled nodes in a mobile edge cloud to support mobile multimedia applications with low latency requirements and dynamically changing VNF placements to address workload changes. Jia et al. [18] and Xi et al. [39] proposed a solution of offloading mobile services with NFV instances in a mobile edge cloud, assuming that all VNFs in a service chain are consolidated into a single edge node. Nam et al. [28] studied the problem of service chaining in a clustered network with mobile edge networks, IoT networks, and backbone networks, with the aim to minimize the average service time of traffic flow. Cziva et al. [8] formulated the VNF placement problem in mobile edges by considering both latency fluctuations of links and mobility of mobile users. Xu et al. [37] investigated a problem of service placement in mobile edge cloud–enabled cellular networks and proposed algorithms based on Lyapunov optimization and Gibbs sampling to reduce computation latency for end users. These studies did not consider the sharing of existing VNF instances in multi-tier cloud networks. They thus may not be suitable for NFV-enabled service chaining provisioning in multi-tier cloud networks.

There are also a few studies on provisioning NFV-enabled network services for IoT applications, which focus on light-weight virtualization techniques, architecture design, or application provisioning [3, 9, 13, 27]. For example, to enable VNFs running in IoT gateways, light-weight implementation of VNFs is required, and several studies have focused on developing efficient methods and implementations of VNFs in IoT gateways [3, 9, 13]. Yu et al. [42] studied the problem of IoT application provisioning with the objective of meeting computing, network bandwidth, and QoS requirements of IoT applications. However, they do not consider the processing of IoT traffic by VNFs. Mouradian et al. [27] proposed an architecture of NFV- and SDN-based distributed IoT gateways for large-scale disaster management. However, joint VNF placement and chaining for service chains required by IoT applications are not the focus of those studies, as they do not consider

multi-tier mobile cloud network either. However, Zhang et al. [43] investigated the problem of placing VNFs in both edge and cloud servers by providing adaptive and efficient methods for 5G network slices. Yet their methods do not consider the mobility of IoT nodes. Farhadi [11] studied the problem of service placement/migration and request scheduling in a two-tier cloud network by proposing a constant-factor approximation algorithm for the problem. However, they do not consider the chaining of services for IoT applications. Novel methods and algorithms are needed for IoT service provisioning in NFV-enabled two-tier cloud networks.

3 PRELIMINARIES

In this section, we first introduce the system model and notations, and then we define the problems precisely.

3.1 System Model

We consider a two-tier cloud network consisting of a mobile edge cloud in a wireless metropolitan area network (WMAN) with cloudlets being deployed in its APs and a distributed cloud that consists of distributed data centers. IoT applications, such as smart home, smart city, and smart grid, are being deployed in the two-tier cloud network to consistently process their user traffic. Let $G = (V \cup CL \cup \mathcal{DC}, E)$ be the two-tier cloud network, where V is a set of switches, CL is a set of resource-constrained cloudlets that are located in the WMAN and within the proximity of users, and $\mathcal{DC}$ is a set of resource-rich data centers in the distributed cloud. The cloudlets and data centers can be interconnected by Internet paths, virtual private network (VPN) tunnels, or high-speed fronthaul/backhaul fiber. Here we use an edge $e \in E$ to represent such a path for simplicity. Data transmissions along each edge $e \in E$ will incur a transmission delay.

A mobile edge cloud network lies in Tier 1 of network G, which consists of IoT devices, APs, switches in V, and cloudlets in CL. IoT devices access network G via APs. Switches are used to interconnect the APs and cloudlets, and some of them are attached with cloudlets in CL. Each cloudlet $cl_m \in CL$ has a few servers with an accumulative amount of available computing resources to host various network applications and implement network functions in VMs. Each cloudlet cl_m has a computing capacity B_{cl_m}.

A distributed cloud in Tier 2 is composed of several data centers in $\mathcal{DC}$ that are located in the core network and provide network services to users. Each data center $DC_j \in \mathcal{DC}$ usually has abundant computing resources to implement VMs for network functions. We assume that a number of VNFs are already instantiated at each data center, which can be viewed as resource pools that are provided by most data centers with abundant resources for reducing the time spent waiting for available resources. Figure 1 illustrates an example of the multi-tier cloud network for IoT applications with cloudlets and remote data centers.

3.2 NFV-Enabled User Requests, Service Chains, and Value-Added Network Functions of IoT Applications

IoT applications usually are deployed to sense physical environments continuously and transfer the sensed data to IoT services to clouds for processing. We consider such requests of data transfer as a *user request*. To guarantee the security of data transfer, each user request demands a *service chain* consisting of a sequence of network functions, such as firewalls and intrusion detection systems (IDSes) to process its data, as shown in Figure 2. Denote by r_k a user request and $\mathcal{SC}_k$ the service chain of r_k. Let ρ_k be the volume of data that request r_k needs to transfer from its nearby AP (i.e., a switch in V) to its destination (i.e., service locations like cloudlets or data centers). Denote by s_k and t_k the source and destination of the data transfer of r_k. Assume that there is a set $\mathcal{F}$ of VNFs in the two-tier cloud network G. Let f_l be a type of VNF in $\mathcal{F}$ with $1 \leq l \leq |\mathcal{F}|$. A type-$l$ network

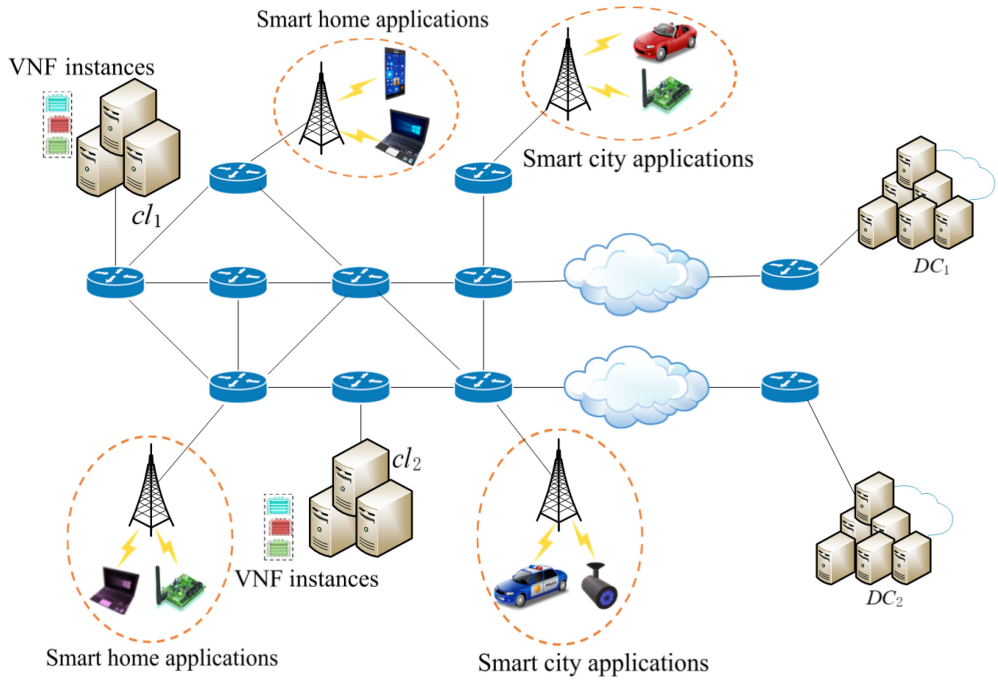


Fig. 1. An example of the multi-tier cloud network for IoT applications with cloudlets and remote data centers.

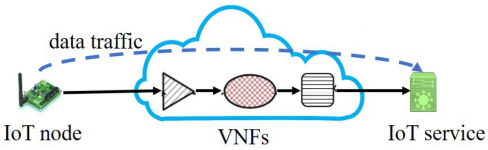


Fig. 2. An example of a service chain for IoT services.

function f_l may be implemented in a cloudlet $cl_m \in CL$ or a data center $DC_j \in \mathcal{DC}$, and each of such implementations is termed as *an instance* of VNF f_l. Without loss of generality, we assume that the amount of computing resource allocated to the VNF instance of f_l is to guarantee its maximal packet processing rate μ_l. Denote by RC^{unit} the amount of resource assigned to process a unit packet rate, and the amount of computing resource needed by an instance of f_l is $\mu_l \cdot RC^{unit}$.

Since each data center in the distributed cloud usually has abundant computing resource, we assume that each data center has instantiated some VNF instances of each f_l already. Denote by n_{lj} the number of existing VNF instances of network function f_l in data center DC_j. Instead, considering that the computing resources of cloudlets are very limited compared with those of data centers, we assume that each cloudlet does not have pre-instantiated VNF instances to avoid resource wastage. However, with the finished requests leaving the system, the idle VNFs in a cloudlet may not be destroyed immediately. Instead, they can be shared by later requests. However, for the sake of security concerns, in this work we prevent VNF sharing among concurrent flows.

Besides conventional network functions, such firewalls and IDSes, we also consider a special set of network functions particularly for IoT applications. Specifically, the order of VNFs in a service chain usually is specified by specific use cases and network services, according to their

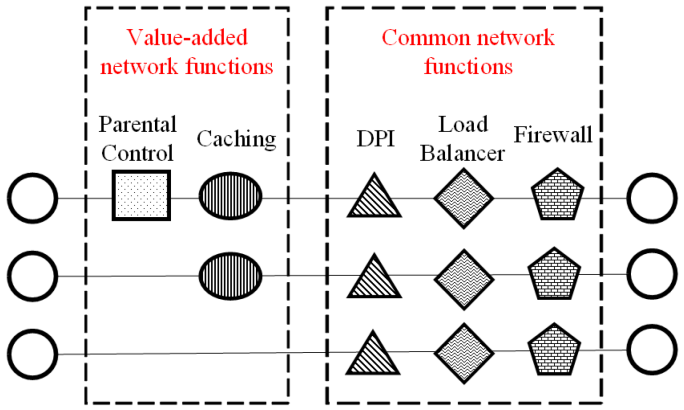


Fig. 3. An illustration of service chains with value-added services.

particular requirements. We observe that IoT applications in the multi-tier cloud network usually have many value-added services, such as VoIP, parental control in smart home applications, and data preprocessing in environment sensing applications. Such *value-added network functions* receive user traffic directly before being forwarded to services in remote data centers for processing. For example, video streaming services usually receive user video traffic and optimize the service before forwarding the processed video streaming for storage in remote data centers [4]. For content providing services, some caching services are used to cache the data that might be used by users [10]. In addition, for virtual reality (VR) and video processing services, network functions related to data preprocessing and rendering are usually deployed in locations close to users. It must be mentioned that the locations of value-added network functions usually play a vital role in guaranteeing the performance of IoT applications. For example, placing video preprocessing into remote data centers may lead to prohibitively long delays, since original videos need to transferred via long paths from devices to remote data centers. However, in the side of remote data centers, there are usually conventional network functions for packet inspection, such as IPFiX, firewalls, IPS, and DDoS. It is clear that value-added network functions are different in different network services, although most network services share great similarities in terms of conventional network functions for packet processing. Considering the mentioned observations, we consider a special set of service chains that share the same sequence of the network functions in the side of remote data centers, as shown in Figure 3. Specifically, let $\mathcal{SC}_{max}$ be the longest service chain. All other sequences meet the following requirements:

- Each service chain SC_k is a common subsequence of $\mathcal{SC}_{max}$.
- The last VNF $f_{|SC_k|}$ is the same as the last VNF $f_{|\mathcal{SC}|_{max}}$.

For example, assume that $\mathcal{SC}_{max} = \{f_1, f_2, f_3\}$, and other service chains include $\{f_2, f_3\}$ and $\{f_3\}$.

3.3 The End-to-End Delay Requirements of User Requests

The end-to-end delay experienced by each admitted request r_k includes the upload delay of its data volume to the nearby AP of the user, the packet processing delay in the VNF instance, the instantiation delay by creating a VNF instance in a cloudlet if needed, and the network latency from the source switch s_k of r_k to its destination via the assigned servers for the VNFs in its service chain SC_k. These delays are described as follows.

Packet processing delay. Each instance of VNF f_l is assigned a certain amount of computing resource to guarantee its packet processing rate μ_l. Therefore, the packet processing delay of request r_k in a network function f_l of its service chain $\mathcal{SC}_k$ is proportional to the volume of its traffic. The packet processing delay $d_p(r_k)$ of r_k thus is the accumulative processing delay by all network functions in its service chain $\mathcal{SC}_k$—that is,

$$d_p(r_k) = \sum_{f_l \in \mathcal{SC}_k} \frac{\rho_k}{\mu_l}. \tag{1}$$

Instantiation delay of VNFs. VNFs run as software in VMs or containers, and instantiating a VM or container usually needs time to instantiate the necessary OS or software before creating the instance of a VNF. Without loss of generality, we assume that the instantiation delay of each type of VNF f_l is a constant, which is denoted by $d_{ins,l}$.

Network latency. The VNFs of the service chain $\mathcal{SC}_k$ of r_k can be placed into multiple locations within the network (either cloudlets in the WMAN or some data centers in the distributed cloud). The path that is used to transmit the traffic of r_k thus can be divided into multiple segments. Let $z_{e,k}$ be a binary variable that indicates whether edge $e \in E$ is used to route the traffic of r_k. The network latency $d_t(r_k)$ of r_k is

$$d_t(r_k) = \sum_{e \in E} z_{e,k} \cdot d_e \cdot \rho_k, \tag{2}$$

where d_e is the delay of transmitting a unit amount of data along edge e.

To guarantee the quality of service (QoS) of each user request r_k, its experienced delay is bounded by an *end-to-end delay requirement* D_k. Denote by $w_{l,m,k}$ the indicator variable that indicates whether cloudlet cl_m is used to implement VNF $f_l \in \mathcal{SC}_k$. The end-to-end delay requirement can be formulated by

$$d_p(r_k) + \sum_{f_l \in \mathcal{SC}_k} \sum_{cl_m \in CL} w_{l,m,k} \cdot d_{ins,l} + d_t(r_k) \leq D_k. \tag{3}$$

3.4 The Admission (Implementation) Cost

For each request r_k, its admission cost usually consists of the instantiation cost of creating new instances for VNFs, the traffic transmission cost, and the processing cost in VNFs.

For the instantiation cost, recall that the implementation of the VNFs in its service chain $\mathcal{SC}_k$ can either make use of existing VNF instances in a data center or create new instances in a cloudlet. Therefore, if an existing VNF instance is adopted, its instantiation cost is saved; otherwise, there is a constant instantiation cost $c_{ins,l}$ for each instance of VNF f_l.

For the processing cost, we assume that it is proportional to the volume of data that will be processed. Let $c_{l,m}$ and $c_{l,j}$ be the costs of processing unit volume of data traffic by VNF f_l in cloudlet cl_m and data center DC_j, respectively. The cost of processing r_k by network function f_l in cloudlet cl_m and DC_j thus are $\rho_k \cdot c_{l,m}$ and $\rho_k \cdot c_{l,j}$, respectively.

Denote by $y_{l,j,k}$ the indicator variable indicating whether data center DC_j implements network function $f_l \in \mathcal{SC}_k$. The processing cost of r_k thus is

$$c_p(r_k) = \sum_{f_l \in \mathcal{SC}_k} \left(\sum_{cl_m \in CL} w_{l,m,k} \cdot c_{l,m} \cdot \rho_k + \sum_{DC_j \in \mathcal{DC}} y_{l,j,k} \cdot c_{l,j} \cdot \rho_k \right). \tag{4}$$

The transmission cost of r_k is proportional to the volume that is transmitted along the path from the source node s_k of r_k to its destination node t_k. Let c_e be the cost of transmitting the unit volume of data traffic along edge $e \in E$. Denote by $c_t(r_k)$ the traffic transmission cost of r_k, which

can be calculated by

$$c_t(r_k) = \sum_{e \in E} z_{e,k} \cdot c_e \cdot \rho_k. \tag{5}$$

3.5 Problem Definitions

Given a two-tier cloud network $G = (V \cup CL \cup \mathcal{DC}, E)$ consisting of a mobile edge cloud in a WMAN and a distributed cloud, a set of user requests S with each request r_k requesting to transmit its data from a source s_k to a destination t_k with a given volume of traffic ρ_k, and having an end-to-end delay requirement D_k, assume that the VNFs in the service chain $\mathcal{SC}_k$ of each request $r_k \in S$ can be placed into multiple locations within G (either cloudlets or data centers), and that some instances of each VNF in $\mathcal{F}$ have already been instantiated in each data center $DC_j \in \mathcal{DC}$. We consider the following optimization problems.

Problem 1. The throughput maximization problem in G is to find a schedule of request admissions such that the weighted system throughput—the accumulative data volume of admitted requests—is maximized, whereas the accumulative operational cost of admitted requests is minimized, subject to the computing resource capacity on each cloudlet in CL.

Problem 2. The throughput maximization problem with value-added network functions in G is to find a schedule of request admissions such that the weighted system throughput is maximized, whereas the accumulative operational cost of admitted requests is minimized, subject to the computing resource capacity on each cloudlet in CL.

For the sake of convenience, symbols used in this article are summarized in Table 1.

4 ALGORITHMS FOR THE THROUGHPUT MAXIMIZATION PROBLEM

In the section, we propose an exact solution for the throughput maximization problem by formulating an ILP solution as follows.

4.1 Exact Solution

Recall that the objective of the throughput maximization problem is to maximize the accumulative traffic volume of admitted requests.

Given a set of S of requests, we use a binary decision variable x_k to decide whether request r_k is admitted. For each VNF $f_l \in SC_k$ of request r_k, recall that we use $w_{l,m,k}$ and $y_{l,j,k}$ to indicate whether cloudlet cl_m and data center DC_j are used to implement f_l, respectively. Denote by $q_{v,k}$ a binary indicator variable that shows whether switch $v \in V$ is used to forward the traffic of r_k. Let $\delta(v)$ denote the incident edges of switch node $v \in V$, respectively.

The objective of the throughput maximization problem thus is to

$$\textbf{ILP}: \max \sum_{r_k \in S} x_k \cdot \rho_k \tag{6}$$

subject to the following constraints.

$$\sum_{f_l \in SC_k} \sum_{cl_m \in CL} \sum_{DC_j \in \mathcal{DC}} (w_{l,m,k} + y_{l,j,k}) = x_k \cdot |SC_k|, \text{for each } r_k \in S, \tag{7}$$

$$w_{l,m,k} + y_{l,j,k} \leq 1, \text{for each } r_k \text{ and each of its } f_l \in SC_k, \tag{8}$$

$$q_{v,k} \leq x_k, \text{for each } r_k \text{ and each switch } v \in V, \tag{9}$$

$$\sum_{e \in \delta(v)} z_{e,k} \leq 2 \cdot q_{v,k}, \text{for each } v \in V \text{ and each } r_k, \tag{10}$$

$$\sum_{e \in \delta(s_k)} z_{e,k} = 1, \text{for } s_k \text{ of each } r_k, \tag{11}$$

Table 1. Symbols

Symbols	Meaning
$G = (V \cup CL \cup \mathcal{DC}, E)$	Two-tier cloud network, where V is a set of switches, CL is a set of resource-constraint cloudlets that are located in the WMAN, and $\mathcal{DC}$ is a set of resource-rich data centers in the distributed cloud
e	a link in E
d_e	Delay of transmitting a unit amount of data along edge e
cl_m	Cloudlet in CL
B_{cl_m}	Capacity of computing resource of cloudlet cl_m
DC_j	Data center in $\mathcal{DC}$
r_k	User request
$\mathcal{SC}_k$	Service chain request of r_k
s_k and t_k	Source and destination of the data transfer of r_k
ρ_k	Volume of data that request r_k needs to transfer from its nearby AP (i.e., a switch in V) to its destination (i.e., service locations like cloudlets or data centers)
$\mathcal{F}$	Set of VNFs in the two-tier cloud network G
f_l	Type-l network function
μ_l	Maximal packet processing rate of f_l
RC^{unit}	Amount of resource assigned to process a unit packet rate
n_{lj}	Number of existing VNF instances of network function f_l in data center DC_j
$d_p(r_k)$	Packet processing delay of r_k
$d_{ins,l}$	Instantiation delay of each type of VNF f_l
$z_{e,k}$	Binary variable that indicates whether edge $e \in E$ is used to route the traffic of r_k
$d_t(r_k)$	Network latency of r_k
D_k	End-to-end delay requirement of request r_k
$w_{l,m,k}$	Indicator variable that indicates whether cloudlet cl_m is used to implement VNF $f_l \in \mathcal{SC}_k$
$c_{ins,l}$	Instantiation cost for each instance of VNF f_l
$c_{l,m}$ and $c_{l,j}$	Costs of processing unit volume of data traffic by VNF f_l in cloudlet cl_m and data center DC_j, respectively
$y_{l,j,k}$	Indicator variable indicating whether data center DC_j implements network function $f_l \in \mathcal{SC}_k$
$c_p(r_k)$	Processing cost of r_k
c_e	Cost of transmitting unit volume of data traffic along edge $e \in E$
$c_t(r_k)$	Traffic transmission cost of r_k
S	Set of user requests
$\mathcal{SC}_{max}$	Longest service chain
x_k	Binary indicator variable that indicates whether request r_k is admitted
$\delta(v)$	Incident edges of switch node $v \in V$
$q_{v,k}$	Binary indicator variable that shows whether switch $v \in V$ is used to forward the traffic of r_k
$S_{\mathcal{SC}}$	Set of requests that all have a service chain requirement of $\mathcal{SC}$
$G'_{\mathcal{SC}} = (V'_{\mathcal{SC}}, E'_{\mathcal{SC}})$	Auxiliary graph constructed for each set $S_{\mathcal{SC}}$ of requests
$cl'_{m,l}$ and $cl''_{m,l}$	Two virtual cloudlet nodes for cl_m in auxiliary graph $G'_{\mathcal{SC}}$
$DC'_{j,l}$ and $DC''_{j,l}$	Two virtual data center nodes for DC_j in auxiliary graph $G'_{\mathcal{SC}}$
p_{s_k,cl_m}	Shortest path from s_k to cl_m in network G
$d(\langle x, y\rangle)$	Delay of edge $\langle x, y\rangle$ in auxiliary graph $G'_{\mathcal{SC}}$
$w(\langle x, y\rangle)$	Weight of edge $\langle x, y\rangle$ in auxiliary graph $G'_{\mathcal{SC}}$
$u(\langle x, y\rangle)$	Capacity of edge $\langle x, y\rangle$ in auxiliary graph $G'_{\mathcal{SC}}$
f	Found single-source min-cost unsplittable flow in auxiliary graph $G'_{\mathcal{SC}}$
$G'' = (V'', E'')$	Constructing of the auxiliary graph in algorithm `Appro`

$$\sum_{e \in \delta(t_k)} z_{e,k} = 1, \text{for } t_k \text{ of each } r_k, \tag{12}$$

$$\sum_{f_l \in SC_k} \frac{\rho_k}{\mu_l} + \sum_{f_l \in \mathcal{SC}_k} \sum_{cl_m \in CL} w_{l,m,k} \cdot d_{ins,l} + \sum_{e \in E} z_{e,k} \cdot d_e \leq D_k, \text{for each } r_k, \tag{13}$$

$$\sum_{r_k \in S} \sum_{f_l \in SC_k} \sum_{cl_m \in CL} w_{l,m,k} \cdot \mu_l \cdot RC^{unit} \leq B_{cl_m}, \text{for each cloudlet } cl_m \in CL, \tag{14}$$

$$\sum_{r_k \in S} \sum_{f_l \in SC_k} \sum_{DC_j \in \mathcal{DC}} y_{l,j,k} \leq n_{lj}, \text{for each data center } DC_j \in \mathcal{DC}, \tag{15}$$

$$\sum_{r_k \in S} \left(\sum_{f_l \in \mathcal{SC}_k} \left(\sum_{cl_m \in CL} w_{l,m,k} c_{l,m} \rho_k + \sum_{DC_j \in \mathcal{DC}} y_{l,j,k} c_{l,j} \cdot \rho_k \right) + \sum_{e \in E} z_{e,k} \cdot c_e \cdot \rho_k \right) \leq BT, \text{for each } r_k \in S, \tag{16}$$

$$x_k, w_{l,m,k}, y_{l,j,k}, q_{v,k}, z_{e,k} \in \{0, 1\}, \tag{17}$$

where Constraint (7) indicates that if a request r_k is admitted, all of its VNFs in SC_k will be assigned to some cloudlets or data centers, and Constraint (8) shows that each $f_l \in SC_k$ is assigned to either a cloudlet or a data center. Constraint (9) ensures that if a request r_k is rejected, no switch will be selected to route its traffic. Constraint (10) captures that if a switch $v \in V$ is used to forward the traffic of r_k, at most two of its incident edges will be used to forward its traffic (one for incoming traffic one for outgoing traffic). Otherwise, one incident edge is used to forward both its incoming and outgoing traffic. Constraints (11) and (12) ensure that no traffic goes into source node s_k and no traffic leaves the destination node t_k, respectively. Constraint (13) enforces the end-to-end delay requirement of r_k. Constraint (14) guarantees that the computing capacity of each cloudlet cl_m is not violated. Constraint (15) says that the number of requests that need f_l and are assigned to DC_j should not exceed the number of available instances of f_l in each data center DC_j. Constraint (16) guarantees that the total cost of implementing the requests is no greater than a given budget. Constraint (17) ensures that each of the variables x_k, $w_{l,m,k}$, $y_{l,j,k}$, $q_{v,k}$, and $z_{e,k}$ is a binary variable.

4.2 Heuristic

The basic idea of the proposed heuristic is based on an observation that requests may share the same service chain in terms of VNF type in many network services. For example, most network functions need a firewall and an IDS network function to prevent external security attacks and proactively respond to possible intrusions.

The proposed heuristic classifies requests into different categories. All requests within one category have the same service chain requirement—that is, their service chains have an identical sequence of VNFs. To admit the requests within each category, we transfer the problem in the multi-tier cloud network G into an unsplittable minimum cost multi-commodity flow problem in another auxiliary graph G'. A feasible solution to the latter corresponds to a feasible solution to the former.

We now describe the proposed heuristic algorithm by first constructing the auxiliary graph with edge weights, capacities, and delays. For each service chain $\mathcal{SC}$, let $S_{\mathcal{SC}}$ be the set of requests that all have a service chain requirement of $\mathcal{SC}$. We deal with each of such sets with requests having the same service chain requirement one by one. Specifically, for each set $S_{\mathcal{SC}}$ of requests, we construct

an auxiliary graph $G'_{SC} = (V'_{SC}, E'_{SC})$. To this end, we build a mapping between the network G and the auxiliary graph G'_{SC}, describing node set V'_{SC} and edge set E'_{SC} of G'_{SC}.

Node set V'_{SC}. The node set of G'_{SC} consists of the following types of nodes:

- *Source and destination nodes*: For each request r_k in S_{SC}, we add its source node s_k to V'_{SC}. We also add the destination node t_k of each request in $S(\mathcal{SC}_k)$ and a common virtual source s for all requests in $S(\mathcal{SC}_k)$
- *Virtual cloudlet nodes*: A new instance for f_l may be instantiated in cloudlet $cl_m \in CL$ as long as cl_m has enough resource for the newly created instance. We thus add two virtual cloudlet nodes (i.e., $cl'_{m,l}$ and $cl''_{m,l}$) for each cloudlet cl_m that has enough computing resources to instantiate an instance for f_i
- *Virtual data center nodes*: For each $f_l \in \mathcal{SC}$, there may exist its instances in data centers in $\mathcal{DC}$. Therefore, for each data center $DC_j \in \mathcal{DC}$ that has VNF instances of f_i, we add two virtual data center nodes (i.e., $DC'_{j,l}$ and $DC''_{j,l}$) into V'_{SC}.

The rationale of splitting each cloudlet or data center node into two virtual cloudlets or data centers is to allow moving processing costs and capacities in nodes of the network G into the edges of the auxiliary graph. The edge costs and node costs can be considered in a unified way.

Edge set E'_{SC}. The edges in E'_{SC} can be classified into the following categories:

- *From common source node to source nodes*: There is an edge from the common virtual source s to each source node s_k of $r_k \in S_{SC}$ in the auxiliary graph G'. The weights of such edges are set to zero, and their capacities are set to infinity.
- *From source nodes to candidate cloudlets/data centers*: There is an edge from each source node s_k to each of the virtual data centers or virtual cloudlets for the first VNF $f_1 \in \mathcal{SC}$, to represent the shortest path in the original network G from source s_k to the candidate data center or cloudlet. Let $\langle s_k, cl'_{m,1}\rangle$ and $\langle s_k, DC'_{j,1}\rangle$ be the added edges in V'_{SC} for virtual cloudlet $cl'_{m,1}$ and virtual data center $DC'_{j,1}$, respectively. Clearly, the weight of edge $\langle s_k, cl'_{m,1}\rangle$ is set to the the sum of cost of the shortest path from s_k to cloudlet cl_m in G:

$$w(\langle s_k, cl_{m,1}\rangle) = \sum_{e \in p_{s_k, cl_m}} c_e, \tag{18}$$

  where p_{s_k, cl_m} is the shortest path from s_k to cl_m in network G. The capacity of the edge is set to infinity. The delay of this edge is

$$d(\langle s_k, cl_{m,1}\rangle) = \sum_{e \in p_{s_k, cl_m}} d_e. \tag{19}$$

  Similarly, the weight of edge $\langle s_k, DC'_{j,1}\rangle$ is set by $w(\langle s_k, DC'_{j,1}\rangle) = \sum_{e \in p_{s_k, DC_j}} c_e$. The capacity of edge $(\langle s_k, DC'_{j,1}\rangle)$ is set to infinity. The delay of this edge is $d(\langle s_k, DC'_{j,1}\rangle) = \sum_{e \in p_{s_k, DC_j}} d_e$.
- *Between virtual cloudlets/data centers of each pair*: For each cloudlet cl_m, we then add an edge from $cl'_{m,l}$ to $cl''_{m,l}$. The weight of this edge is set to the total of (1) the cost of instantiating a new instance in cl_m that is amortized to each its resource demand of processing a unit traffic ($\frac{c_{ins,l}}{RC^{unit}}$), and (2) the cost of processing unit data traffic of request r_k (i.e., $w(\langle cl'_{m,l}, cl''_{m,1}\rangle) = \frac{c_{ins,l}}{RC^{unit}} + c_{l,m}$). The capacity of this edge is set to the capacity of packets that can be processed by the available computing resource of cl_m (i.e., $u(\langle cl'_{m,l}, cl''_{m,1}\rangle) = \lfloor \frac{A_{cl_m}}{RC^{unit}} \rfloor$). Its delay is the processing delay (i.e., $d(\langle cl'_{m,l}, cl''_{m,1}\rangle) = d_{ins,l} + \frac{1}{\mu_l}$). Similarly, for each data center $DC_j \in \mathcal{DC}$, we add an edge from $DC'_{j,l}$ to $DC''_{j,l}$. Its weight and capacity

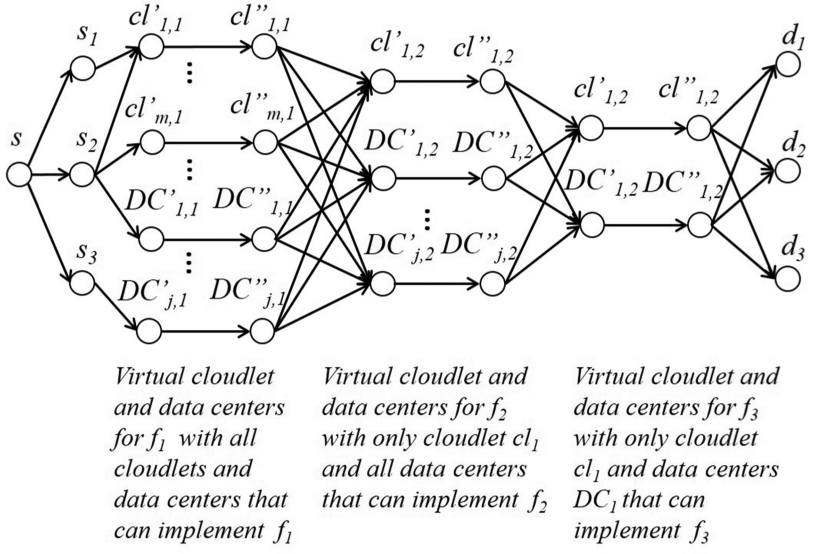


Fig. 4. An example of the auxiliary graph $G'_{SC} = (V'_{SC}, E'_{SC})$.

are set as $w(\langle DC'_{j,l}, DC''_{j,1}\rangle) = c_{l,j}$ and $u(\langle DC'_{j,l}, DC''_{j,1}\rangle) = n_{lj} \cdot \mu_l$, respectively. Its delay is the processing delay (i.e., $d(\langle DC'_{j,l}, DC''_{j,1}\rangle) = \frac{1}{\mu_l}$).

- *Between pairs of virtual cloudlets/data centers*: Since the traffic of each request r_k needs to be processed by VNF instances according to the specified sequence in $\mathcal{SC}$, we then add edges from virtual cloudlets and virtual data centers for f_1 to the virtual cloudlets and data centers for f_2. Specifically, there is an edge from each virtual data center $DC''_{j,1}$ for f_1 to virtual data center $DC'_{j',2}$ or virtual cloudlet $cl'_{m,2}$ for f_2. In other words, $E' \leftarrow E' \cup \{\langle DC''_{j,1}, cl'_{m,2}\}$ and $E'_{SC} \leftarrow E'_{SC} \cup \{\langle DC''_{j,1}, DC'_{j',2}\}$ for all cloudlets in CL that have enough resources to create a new instance for f_2, and all data centers in $\mathcal{DC}$ that have existing instances of f_2. Its weight is set to the transmission cost from DC_j to cl_m or DC'_j in G, its capacity is set to infinity, and its delay is the transmission delay from from DC_j to cl_m or DC'_j in G. In addition, there is an edge from each virtual cloudlet $cl''_{m,1}$ for f_1 to virtual data center $DC'_{j,2}$ or virtual cloudlet $cl'_{m',2}$ for f_2. In other words, $E'_{SC} \leftarrow E'_{SC} \cup \{\langle cl''_{m,1}, cl'_{m',2}\}$ and $E'_{SC} \leftarrow E'_{SC} \cup \{\langle cl''_{m,1}, DC'_{j,2}\}$. Its weight is set to the transmission cost from cl_m to $cl_{m'}$ or DC_j in G, whereas its capacity is set to infinity, and its delay is the transmission delay from cl_m to $cl_{m'}$ or DC_j in G.
- *From the virtual cloudlet/data centers to destination nodes*: There are edges from candidate cloudlets and data centers for the last VNF $f_{|SC|}$ to the destination nodes of all requests in S_{SC}. The weights/delays of such edges are set to the transmission costs/delays of a packet from the location for the last VNF to the destinations in G. Their capacities are set to infinity.

Having constructed the auxiliary graph G'_{SC} (an example of G'_{SC} is shown in Figure 4), we now admit the requests one by one. Specifically, for each request r_k, we find an unsplittable minimum cost flow from s to its destination node in the auxiliary graph G'_{SC}. Let p'_{s,t_k} be the path that is traversed by the found flow. We then check whether p'_{s,t_k} can meet its end-to-end delay requirement. If so, the request is admitted, and the corresponding cloudlets and data centers of virtual cloudlets and data centers in p'_{s,t_k} will be the locations to implement the VNFs in $\mathcal{SC}_k$. If not, we remove the edge in p'_{s,t_k} with the maximum delay from G'_{SC} and continue to find an unsplittable min-cost

ALGORITHM 1: Heu

Input: $G = (V \cup CL \cup \mathcal{DC}, E)$, computing capacity B_{cl_m} for each $cl_m \in CL$, and a set of requests S.
Output: An admission of requests in S, and the locations for the VNFs in service chain SC_k of each admitted request r_k.

for each service chain $\mathcal{SC}$ **do**
  Let $S_{\mathcal{SC}}$ be the set of requests that require service chain $\mathcal{SC}$;
  Construct auxiliary graph $G'_{\mathcal{SC}} = (V'_{\mathcal{SC}}, E'_{\mathcal{SC}})$, as shown in Figure 4;
  for each request $r_k \in S_{\mathcal{SC}}$ **do**
    Let p be the found path for r_k in network G;
    $p \leftarrow \emptyset$;
    Denote by E'_{rem} the set of removed edges due to the violation of request delay requirement;
    $E'_{rem} \leftarrow \emptyset$;
    while $p \neq \emptyset$ **do**
      Add all edges in E'_{rem} into $E'_{\mathcal{SC}}$;
      Find an unsplittable min-cost flow in auxiliary graph $G'_{\mathcal{SC}}$ by invoking the algorithm of Kolliopoulos and Stein [20];
      Let f be the found flow;
      if flow f meets the delay requirement of request r_k **then**
        Replace each of all other edges in f with its corresponding shortest path in network G for each of the admitted request;
        Assign the resulting path to p;
        Add all removed edges in E'_{rem} into $G'_{\mathcal{SC}}$;
      else
        Remove the edge e'_{mDelay} in f that has the highest delay;
        $E'_{rem} \leftarrow E'_{rem} \cup \{e'_{mDelay}\}$;
      end if
    end while
  end for
end for

Let f be the found single-source min-cost unsplittable flow from s to the destinations. If the flow along edge $\langle s, s_k \rangle$ is non-negative, request r_k is admitted; otherwise, it is rejected.
Replace each of all other edges in $G'_{\mathcal{SC}}$ with its corresponding shortest path in network G for each of the admitted request.

multi-commodity flow. The procedure continues until its delay requirement is met. Otherwise, the request is rejected.

The preceding procedure continues until all requests that require each type of service chain are all considered, as shown in Algorithm 1.

4.3 Discussion on Affinity, Security, and Policy Requirements of VNFs

In the proposed heuristic, we assumed that a VNF of a service chain can be placed into any cloudlet or data center with sufficient computing resources. In real VNF deployments, there usually are some affinity, security, and policy requirements. For example, some VNFs rely on GPU or FPGAs to accelerate the their packet processing. GPU-based packet processing accelerations are normally used to implement IDSes. Such VNFs thus prefer to stay in cloudlets or data centers with such acceleration supports. Their placements therefore need to consider the *affinity* requirement. However, for the sake of security reasons, user requests may specify a given set of locations that can place their VNFs.

The proposed heuristic algorithm Heu can be easily extended to consider this scenario. Specifically, assume that each request r_k has a set of specified cloudlets/data centers that can meet its affinity, security, and policy requirements. We still need to construct the auxiliary graph for each service chain. Each request may have different sets of specified cloudlets or data centers. For each request r_k, we then adjust the constructed auxiliary graph by removing the edges in G'_{SC} with endpoints that are not in its specified set.

4.4 Algorithm Analysis

We now analyze the solution feasibility and the performance of the proposed algorithm.

Lemma 4.1. *Algorithm 1 delivers a feasible solution to the throughput maximization problem in G, by meeting the end-to-end delay requirement of each admitted request while the resource capacity violation of each cloudlet is at most* $|\mathcal{SC}_{max} - 1| \cdot \arg\max_{f_l} \mu_l \cdot RC^{unit}$, *where* $\mathcal{SC}_{max}$ *is the longest sequence of service chain.*

Proof. We first show that the end-to-end delay requirement of each admitted request is met. In the proposed algorithm, we first find an unsplittable min-cost multi-commodity flow from s to its destination node t_k in the auxiliary graph G'_{SC}. Let p'_{s,t_k} be the path that is traversed by the found flow. We then check whether p'_{s,t_k} can meet its end-to-end delay requirement. If not, the algorithm will exclude the edge that has the highest delay in p'_{s,t_k} and find the shortest path again. The procedure continues until the end-to-end delay requirement is met or the request is rejected. Therefore, as long as the request is admitted, its end-to-end delay requirement is met.

We then show that the capacity of each cloudlet is violated by at most $|\mathcal{SC}_{max} - 1| \cdot \arg\max_{f_l} \mu_l \cdot RC^{unit}$. For each request, there are $|\mathcal{SC}_k|$ VNFs in its service chain, and all of its VNFs can be implemented in cloudlet cl_m. This means that in the auxiliary graph G'_{SC}, edges $\langle cl'_{m,1}, cl''_{m,1}\rangle$, $\langle cl'_{m,2}, cl''_{m,2}\rangle, \ldots$, and $\langle cl'_{m,|\mathcal{SC}_k|}, cl''_{m,|\mathcal{SC}_k|}\rangle$ will all be traversed by the unsplittable minimum cost flow. As shown in Figure 4, the capacity of each edge $\langle cl'_{m,l}, cl''_{m,l}\rangle$ is set to the number of packets that can be processed by the available computing resource of cl_m (i.e., $\lfloor \frac{A_{cl_m}}{RC^{unit}} \rfloor$). If the first VNF in SC_k saturates cloudlet cl_m, all of the remaining VNFs will not have enough computing resources. In the worse case, the computing resources of cl_m can be violated by at most $|\mathcal{SC}_{max} - 1| \cdot \arg\max_{f_l} \mu_l \cdot RC^{unit}$. □

Theorem 4.2. *Given a two-tier cloud network* $G = (V \cup CL \cup \mathcal{DC}, E)$, *a set S of user requests with each having a service chain requirement* SC_k, *and an end-to-end delay requirement,* Algorithm 1 *delivers a feasible solution for the throughput maximization problem in G within* $O(|S| \cdot (|V| + |\mathcal{DC}| + |CL|)^5)$ *time.*

Proof. Since the feasibility of the solution is shown in Lemma 4.1, here we analyze the running time of the proposed heuristic as follows.

As shown in Algorithm 1, it consists of two stages: (1) constructing the auxiliary graph and (2) finding an unsplittable minimum flow that meets the delay requirement for each request. For stage (1), it is clear that the time of constructing of the auxiliary graph G'_{SC} depends on the number of edges in G', which is $O(|V'|^2)$. We can see that data center and cloudlet nodes are duplicated into pairs of virtual data center and cloudlet nodes. Thus, $|V'_{SC}| = O(|V| + |\mathcal{DC}| + |CL|)$. The running time of the first stage thus is $O((|V| + |\mathcal{DC}| + |CL|)^2)$. For stage (2), according to the algorithm of Kolliopoulos and Stein [20], the running time of finding an unsplittable minimum cost flow is $O(|V'_{SC}| \cdot |E'_{SC}|)$. Since the found the path may not meet the delay requirement, the finding of an unsplittable flow may be repeated at most $O(|E'_{SC}|)$ times. Part (2) thus takes $O(|V'_{SC}| \cdot |E'_{SC}|^2)$ time.

Consider that the auxiliary graph is constructed for each type of service chain, and there are at most $|S|$ types of service chains (i.e., each request having a different service chain). The running time of the algorithm thus is $O(|S| \cdot |V'_{SC}|^2 + |S| \cdot |V'_{SC}| \cdot |E'_{SC}|^2) = O(|S| \cdot |V'_{SC}|^5) = O(|S| \cdot (|V| + |\mathcal{DC}| + |CL|)^5)$.

It must be mentioned that the time complexity is based on the worst-case analysis by assuming that $|E'_{SC}| = |V'_{SC}|^2$ and each request has a different service chain. However, we can see that the constructed auxiliary graph G'_{SC} is never fully connected. In addition, there usually is a constant number of types of service chain. The running time of the proposed algorithm thus can be represented as $O(|V'_{SC}|^2 + |V'_{SC}| \cdot |E'_{SC}|^2)$ as well. □

5 AN APPROXIMATION ALGORITHM FOR THE THROUGHPUT MAXIMIZATION PROBLEM WITH VALUE-ADDED NETWORK FUNCTIONS

In the section, we propose an approximation algorithm for the throughput maximization problem with value-added network functions.

5.1 Overview of the Algorithm

We aim to propose an approximation algorithm for the throughput maximization problem with value-added network functions. One fundamental challenge of devising an approximation algorithm for the problem is how to admit the requests in S concurrently, considering that different requests may have different service chain requirements. To this end, we introduce a novel graph transformation technique that constructs an auxiliary graph $G'' = (V'', E'')$, based on the constructed auxiliary graph G'_{SC} in the previous section. We then reduce the problem into the problem in two-tier cloud network G of a single-source min-cost multi-commodity problem in G'' that concurrently admits all requests in S. An approximate solution to the latter will return an approximate solution to the former.

5.2 Approximation Algorithm

We start by constructing the auxiliary graph $G'' = (V'', E'')$. For service chain $\mathcal{SC}_{max}$, we follow the construction of auxiliary graph G'' in the previous section. We then add the source nodes for each service chain $\mathcal{SC}_k$ into V''. Each of such source nodes is connected to the corresponding virtual cloudlet/data centers for its first VNF, which may not be necessary the first VNF in $\mathcal{SC}_{max}$. Their destination nodes are also included into V'', and the virtual cloudlet/data center nodes are connected to them. Edge weights and capacities are set according to similar edges in G'_{SC}. Figure 5 illustrates an example of the constructed auxiliary graph.

We then reduce the original problem in the two-tier cloud network G to a problem of finding a single-source unsplittable min-cost flow problem in the constructed auxiliary graph G''. Specifically, we assume that each request corresponds a *commodity* that needs to transfer a demand of ρ_k from the common source node s to its destination t_k in G''. Each of such transfers needs to be done via a single path in G'', and the edge capacities should be reserved. Clearly, the solution to the single-source unsplittable min-cost multi-commodity flow problem in G'' will return a feasible solution to the original problem. The approximation algorithm is shown in Algorithm 2.

5.3 Algorithm Analysis

Lemma 5.1. Algorithm 2 *obtains a feasible solution for a special case of the throughput maximization problem with value-added network functions, which meets the service chain requirement SC_k of each admitted request r_k while the computing capacity violation of each cloudlet is upper bounded by $(|\mathcal{SC}_{max}| - 1) \cdot \arg\max_{f_l} \mu_l \cdot RC^{unit}$.*

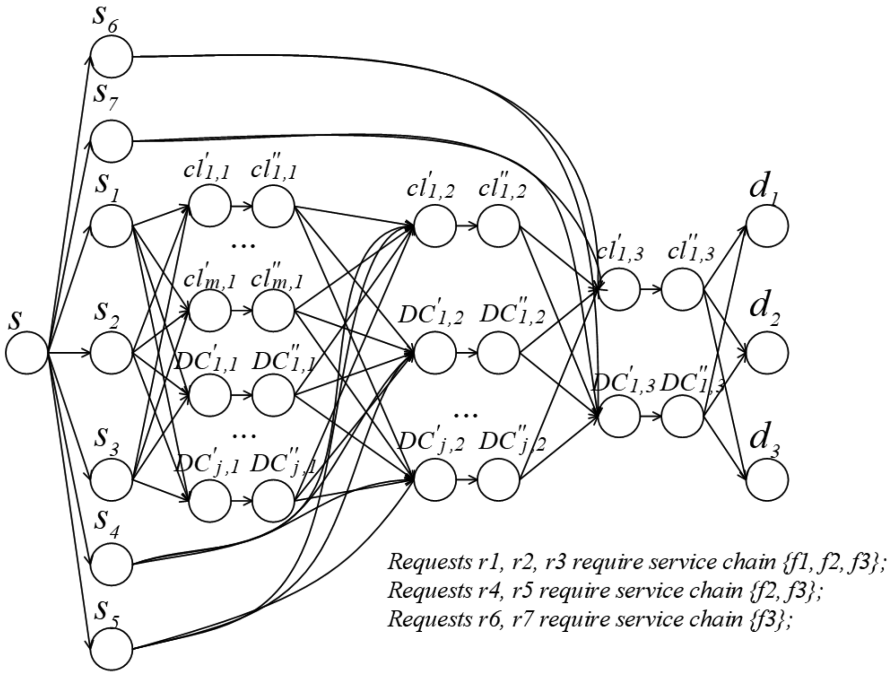


Fig. 5. An example of the auxiliary graph $G'' = (V'', E'')$.

ALGORITHM 2: Appro

Input: $G = (V \cup CL \cup \mathcal{DC}, E)$, computing capacity B_{cl_m} for each $cl_m \in CL$, and a set of requests S.

Output: An admission of requests in S, and the locations for the VNFs in service chain SC_k of each admitted request r_k.

1: Construct an auxiliary graph $G'' = (V'', E'')$, as shown in Figure 5;
2: Find a single-source min-cost unsplittable flow in auxiliary graph G'' by invoking the algorithm of Kolliopoulos and Stein [20];
3: Let f be the found single-source min-cost unsplittable flow from s to the destinations. If the flow along edge $\langle s, s_k \rangle$ is non-negative, request r_k is admitted; otherwise, it is rejected;
4: Replace each of all other edges in G'' with its corresponding shortest path in network G for each of the admitted request.

Proof. We first show that the resource capacity of each cloudlet is violated by at most $|\mathcal{SC}_{max}| - 1$. The difference between Algorithm 1 and Algorithm 2 is that the latter schedules the requests with the same service chain requirement concurrently, by considering each request as a commodity in the single-source min-cost unsplittable flow problem. In the construction of the auxiliary graph G'', it can be seen that the capacities of cloudlets are moved to edge capacities in G'' that are reserved by a feasible single-source min-cost multi-commodity unsplittable flow [20]. Therefore, the capacity of each edge in G'' is not violated for the concurrent admissions of admitting multiple requests. Instead, as shown in Lemma 4.1, using the same cloudlet for multiple VNFs of a service chain can violate the capacity of a cloudlet.

We then show that the service chain requirement is met. This is guaranteed by connecting each source node in G'' to the corresponding virtual cloudlet/data centers for its first VNF, which may not necessary be the first VNF in $\mathcal{SC}_{max}$. □

Theorem 5.2. *Given a two-tier cloud network $G = (V \cup CL \cup \mathcal{DC}, E)$, and a set of user requests, Algorithm 1 delivers a feasible solution in $O(|S| \cdot (|V| + |\mathcal{DC}| + |CL|)^5)$ time for the throughput maximization problem with value-added network functions. The approximation ratio of the proposed algorithm is $0.075 - \epsilon$, where ϵ is an accuracy parameter in the single-source min-cost unsplittable flow problem with $0 < \epsilon < 0.075$.*

Proof. Since the feasibility of the solution is shown in Lemma 5.1, here we only analyze the approximation ratio and the running time of the algorithm.

To show the approximation ratio, we need to show that the solution to the single-source min-cost unsplittable flow problem in the auxiliary graph corresponds to a feasible solution to the throughput maximization problem with value-added network functions in original network G. Let f'' be such a flow in auxiliary graph G''. In the construction of auxiliary graph G'', it can be seen that there is an edge from s to each of the source nodes of the requests, and there is edge from the locations for the last VNF in $\mathcal{SC}_{max}$ to each of the destinations of the requests. For each request r_k, it thus corresponds to the transmission of its traffic from its source s_k to its destination t_k. If there is a non-negative flow in edges $\langle cl'_{m,l}, cl''_{m,l}\rangle$ or $\langle DC'_{j,l}, DC''_{j,l}\rangle$, it corresponds to the assignment of VNF f_l to cl_m or DC_j. Clearly, the solution derived from f'' is a feasible solution to the throughput maximization problem with value-added network functions. Since the approximation ratio of the single-source min-cost unsplittable flow problem is $0.075 - \epsilon$, the approximation ratio of algorithm 2 is $0.075 - \epsilon$ as well.

Here we analyze the running time of the approximation algorithm. Algorithm 2 consists of two parts: (1) constructing the auxiliary graph and (2) finding a single-source min-cost unsplittable minimum flow. For part (1), it is clear that the time of constructing of the auxiliary graph G'' depends on the number of edges in G'', which is $O(|V''|^2)$. We can see that data center and cloudlet nodes are duplicated into pairs of virtual data center and cloudlet nodes. Thus, $|V''| = O(|V| + |\mathcal{DC}| + |CL|)$. The running time of the first part thus is $O((|V| + |\mathcal{DC}| + |CL|)^2)$. For part (2), according to the algorithm of Kolliopoulos and Stein [20], the running time of finding an unsplittable minimum cost flow is $O(|V'| \cdot |E'|)$. Since the found path may not meet the delay requirement, the finding of an unsplittable flow may be repeated at most $O(|E'|)$ times. Part (2) thus takes $O(|V'| \cdot |E'|^2)$ time. □

6 ONLINE AND LEARNING-BASED ALGORITHM FOR THE THROUGHPUT MAXIMIZATION PROBLEM WITH MOBILE AND ADAPTIVE-RATE IOT DEVICES

Thus far, we have assumed that mobile users are stationary and do not switch their connected APs prior to finishing the implementations of their requests. We have also assumed that the packet rate ρ_k of each request r_k is given and fixed. However, IoT devices may have adaptive packet rates considering that their energy is constrained. In this section, we consider the throughput maximization problem by removing these assumptions.

Recall that each request r_k transfers its traffic from a nearby AP (a.k.a. its source s_k) to its destination t_k for processing. Therefore, when the IoT device of r_k moves, the source node of r_k can change as well. Its packet rate ρ_k can also change, considering that it may reduce its packet rate when the residual energy is low. In the following, we first propose an algorithm that proactively places VNFs of requests by considering the mobility and adaptive rates of IoT devices. Then, we devise an online algorithm that smartly migrates the placed VNFs while the IoT devices moves such that the QoS experienced by the admitted requests are met.

6.1 Online and Learning-Based Algorithm

Let $\mathcal{S}_k$ be the set of historical APs that request r_k has registered. For each $s_k \in \mathcal{S}_k$, request r_k may transfer a portion of its data to the two-tier cloud network. Since the traffic flow is not splittable,

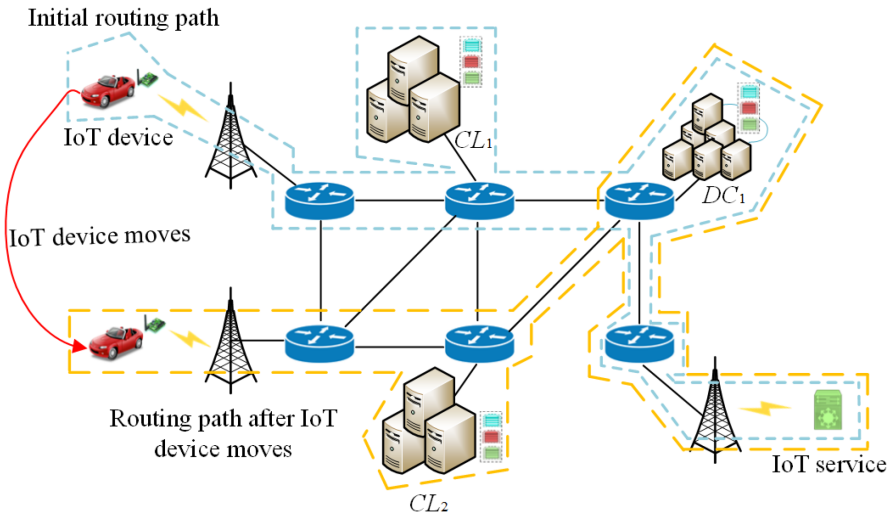


Fig. 6. An example of VNF relocation for mobile IoT devices.

we assume that all different portions of the data are still processed by the same service chain. However, we allow the data traffic to be forwarded to another instances of its service chain for processing when the IoT device of the request moves to a new place. Such an example is shown in Figure 6.

The basic idea of the proposed dynamic VNF placement algorithm is to find a set of candidate placement of VNFs of r_k by considering its mobility, and to find a placement with the minimum cost as its *initial placement*. To this end, we consider each request r_k as $|\mathcal{S}_k|$ *virtual requests*, with each virtual request $r_{k,l}$ with a source node $s_k \in \mathcal{S}_k$, where $1 \leq l \leq |\mathcal{S}_k|$. To find the initial placement for the service chain of r_k, we invoke `Algorithm 1` for $|\mathcal{S}_k|$ times to determine the source node of r_k that can achieve the minimum cost. Specifically, we consider $S = \{r_{k,l} \mid 1 \leq l \leq \mathcal{S}_k\}$ as the input of `Algorithm 1`. We then select the paths (for the virtual requests) with the minimum cost and find the source of the path after invoking `Algorithm 1`. Let s_k be the found source of the path. We then use the selected source for request r_k and its corresponding placement as the initial placement of VNFs of r_k.

Notice that the location of an IoT device of each request may change over time. In addition, the packet rate of each request changes with the energy statuses of IoT devices. The initial placement of the service chain of request r_k may need to be adjusted if the IoT device moves to another location and registers to a different AP or its request changes packet rate. The reason is that the initial placement of its VNFs may no longer meet its delay requirement. We observe that this may create a high overhead if a new service chain will be created for every movement of the IoT device of the request. Here we adopt a proactive approach that predicts both the next AP (i.e., source node) to which the IoT device of request r_k will move and its packet rate. In the following, we describe the prediction methods for mobility and adaptive rates, respectively.

We first describe the prediction method for the next moves of IoT devices. In 5G networks, since APs are densely deployed in an area, the IoT device of each request may within multiple AP coverage ranges. If this is the case, the IoT device chooses an AP to connect, which is determined by its energy status and its communication channel quality with the AP. We consider this as a blackbox and predict its behavior via a reinforcement learning (RL) process, which is formulated in the following.

- *Reward*: In the throughput maximization problem, we aim to maximize the system throughput while meeting the delay requirements of requests. We assume that there is an agent for each admitted request. Since the request of an agent is already admitted, its objective is to minimize the cost of implementing its request while meeting its delay requirement, considering the mobility of its IoT device. The reward of each agent for choosing an action thus is defined as the the cost reduction due to the selected action.
- *State space*: The state of the system consists of current VNF placements of each request and its current AP (i.e., source node). Specifically, the state information includes (1) the locations of placed VNFs of each request, (2) the new AP of the request, (3) the resource availabilities of the two-tier cloud network, and (4) the experienced delays of current admitted requests.
- *Action space*: The agent intelligently observes the mobility of each request and adaptively adjusts its sensibility to the moves of each request. Specifically, for each request, its IoT device may follow a fixed pattern of mobility, such as speed and path. If the agent is sensible to every subtle move of the request, the VNFs of the request may be migrated frequently, thereby increasing the cost of implementing the request. However, if the agent is insensible, the delay requirement of the request could be violated. Therefore, the RL procedure adaptively adjusts the sensibility of the agent. Let γ be the sensibility of an agent, which is considered as the probability of adjusting VNF placements of r_k when its IoT device switches to another AP. Therefore, we have $0 \leq \gamma \leq 1$. Thus, the agent of each admitted request needs to decide whether to react to a move of the IoT device of each request by adjusting its sensibility γ. Specifically, the action taken for the agent can be modeled as $\{-1, 0, 1\}$, where -1 means that the agent wishes to increase its sensibility by a step ϵ, 0 indicates that the agent wants to maintain its current sensibility, and 1 implies that it wants to decrease its sensibility. For each move of the IoT device of r_k, the agent needs to select an action such that its sensibility is determined. It then adjusts the placements of VNFs of r_k with probability γ by invoking `Algorithm 1`. If the cost of implementing a request increases by a pre-defined threshold ϑ or its delay increases by another threshold θ, this means that the environment may suffer from great changes. We allow the agent to choose its actions randomly.

We then proceed with the prediction method for the packet rates of IoT devices. A popular approach of promoting the energy efficiency of IoT devices is to adaptively adjust the data collection rate of each IoT device, which implies that less data can be collected if the IoT device is at its low-battery level. Therefore, when an IoT device of request r_k moves around, the volume of data ρ_k of the request may change as well. Since the focus of the work is not about data collection of IoT devices, we assume that ρ_k is not given and needs to be predicted. To determine the amount of data of each request, we make use of historic request traces to predict the data volume changes while the request is moving. Specifically, we adopt an auto-regression mechanism to predict the data volume $\hat{\rho}_k(m)$ of request r_k at its next move, using the information of the its previous p moves, assuming that the value of p is given:

$$\hat{\rho}_k(m) = a_1 \cdot \rho_k(m-1) + a_2 \cdot \rho_k(m-2) + \cdots + a_p \cdot \rho_k(m-p), \tag{20}$$

where $a_{p'}$ is a constant with $0 \leq a_{p'} \leq 1$, $\sum_{l=1}^{p} a_l = 1$, and $a_{p_1} \geq a_{p_2}$ if $p_1 < p_2$. The detailed algorithm is shown in Algorithm 3.

7 PERFORMANCE EVALUATION THROUGH SIMULATIONS

In this section, we evaluate the performance of the proposed algorithms through experimental simulation. We also investigate the impact of important parameters on the performance of the proposed algorithms.

ALGORITHM 3: Dynamic_Heu

Input: $G = (V \cup CL \cup \mathcal{DC}, E)$, computing capacity B_{cl_m} for each $cl_m \in CL$, and a set of requests S.
Output: An admission of requests in S, and the locations for the VNFs in service chain SC_k of each admitted request r_k.

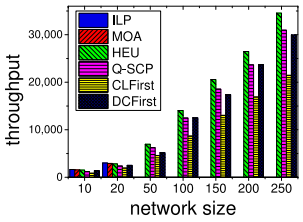
/*Initial placement of VNFs of each request*/;
Consider each request r_k as $\mathcal{S}_k$ virtual requests;
for each r_k **do**
    Invoke Algorithm 1 by assuming that $S = \{r_{k,l} \mid 1 \leq l \leq \mathcal{S}_k\}$;
    Let $\mathcal{P}$ be the returned results, and p_{min} be the placement with the minimum cost;
    Consider the source node in placement p_{min} as the source of r_k;
    Reset the resource availabilities of G to its initial states;
end for
Find the placement with the minimum cost among the $|\mathcal{S}_k|$ placements of virtual requests of r_k;
/*RL-assisted dynamic adjustment of placed VNFs*/;
while IoT device of a request r_k moves **do**
    Predict the data volume of r_k by Equation (20);
    With probability γ the agent replaces the VNFs of r_k;
    Calculate the cost of replacement and delay according to the predicted data volume $\hat{\rho}_k$;
    if the cost of replacement is greater than ϑ or the delay is higher than θ **then**
        Randomly select an action from $\{-1, 0, 1\}$ as its next action.
    end if
end while

7.1 Environment Settings

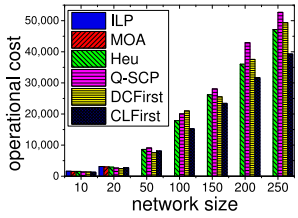
We consider multi-tier cloud networks with sizes varying from 50 to 250 switch nodes and around five data centers, where each network topology is generated using GT-ITM [33]. The number of cloudlets in the mobile edge network is set to 10% of the network size, and they are randomly co-located with switches in the network edge. We also use real network topologies, such as an ISP network from Spring et al. [31]. The computing capacity of cloudlet varies from 40,000 to 120,000 MHz with around tens of servers [1]. Following existing studies [12, 25], we consider five popular types of conventional network functions, such as DPI, load balancer firewall, and NAT. For value-added network functions, we consider parental control and caching. For the sake of diversity, the VNF sequence of each service chain is randomly generated. Notice that value-added network functions are always placed before conventional network functions in a service chain. The data of each request is randomly drawn from [50, 200] MB, and its delay requirement is randomly drawn from [0.5, 5] seconds. The running time of each algorithm is obtained based on a machine with a 3.70-GHz Intel i7 hexa-core CPU and 16 GiB of RAM. Unless otherwise specified, these parameters will be adopted in the default setting.

We compare the performance of the proposed algorithms with the following benchmark algorithms:

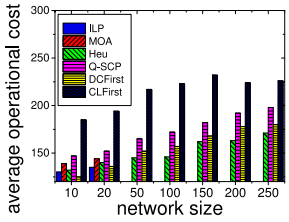
- Since we assume that the VNFs of each request may be placed to multiple cloudlets, we compare our solutions with existing solutions in Vizarreta et al. [34] that consolidate VNFs into a single location, which is referred to as Q-SCP. For each request r_k, Q-SCP first finds a routing path and selects candidate cloudlets or data centers on the routing path that can meet the delay requirement of r_k. Then, for each SC_k, it calculates the minimum cost for implementing SC_k for each candidate node and selects the node with the lowest cost to implement the instance of f_l.



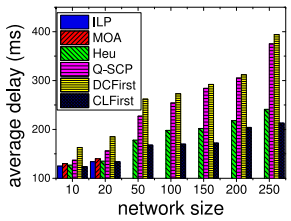
(a) System throughput.



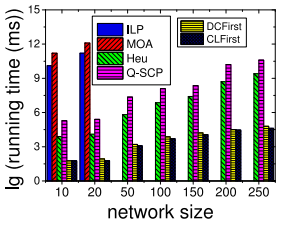
(b) Total operational cost.



(c) Average operational cost per request.



(d) Average delay per request.



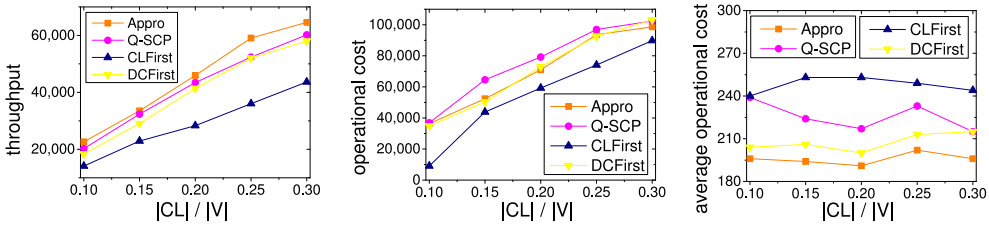
(e) Running time.

Fig. 7. The performance of the ILP, MOA, Heu, Q-SCP, CLFirst, and DCFirst algorithms.

- The second benchmark we adopt is a multi-objective-based optimization algorithm in Addis et al. [2], which is referred to as MOA. It first formulates the problem into multi-objective MILP and then transforms the problem into a single-objective optimization problem to solve.
- We also use a greedy solution that prefers to select existing VNF instances in data centers for each request r_k as our benchmark. Specifically, it finds the data center that is closest to source node s_k and has a VNF instance for its first VNF in SC_k, if such a data center does not exist, a new VNF instance in its closest cloudlet of the mobile edge cloud is created. The procedure continues until all VNFs in SC_k are considered, which is referred to as algorithm DCFirst.
- Another greedy benchmark prefers to place VNFs in the cloudlets of the mobile edge by first instantiating new instances for VNFs of each request until all cloudlets are saturated and then choosing existing VNF instances in data centers. This algorithm is referred to as CLFirst.

7.2 Performance Evaluation of the ILP and Heu Algorithms

We first evaluate the performance of algorithms ILP, MOA, Heu, Q-SCP, CLFirst, and DCFirst, in terms of the system throughput, operational cost, average operational cost of each request, average delay experienced of each request, and the running time, by varying the number of switches in the two-tier cloud network from 50 to 250. The results are shown in Figure 7, from which we can see that the ILP algorithm delivers the highest throughput for network sizes 10 and 20. However, it may not deliver an exact solution when the network size is larger than 20 in a reasonable amount of time, due to its poor scalability. Similarly, we can see from Figure 7(a) and (c) that the MOA algorithm delivers a higher throughput than other algorithms except ILP, because it aims to find an optimal solution via solving the MILP. Yet as can be seen from Figure 7(e), algorithm MOA cannot deliver an optimal solution when the network size is no less than 20. From Figure 7(a), we can also see that the Heu algorithm achieves a higher throughput than those of the Q-SCP, CLFirst, and



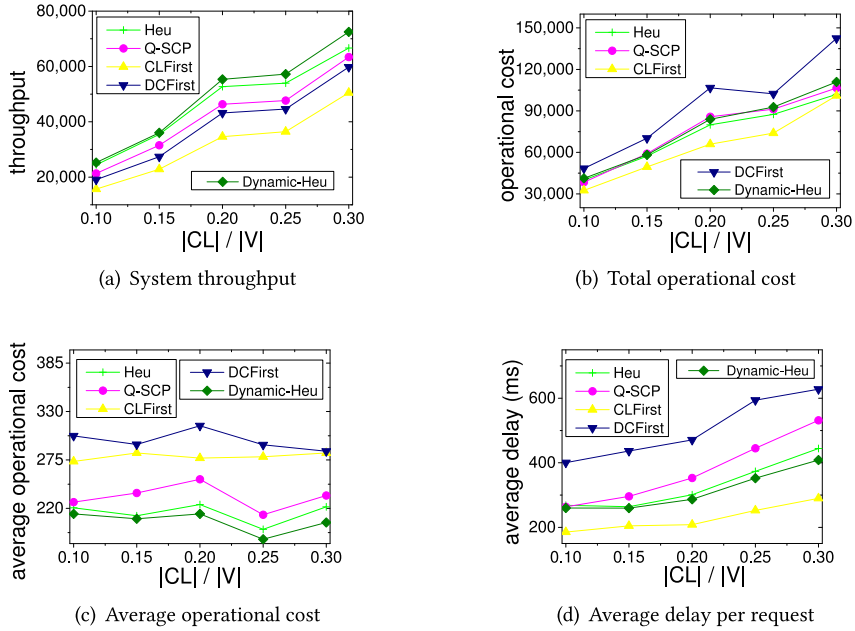
(a) System throughput. (b) Total operational cost. (c) Average operational cost per request.

Fig. 8. The performance of the Appro, Q-SCP, CLFirst, and DCFirst algorithms in network AS1755.

DCFirst algorithms. The reason is that the Heu algorithm places the VNFs in each service chain to multiple locations (cloudlets or data centers), and this allows the workload of each cloudlet to be balanced, thereby reducing small resource slices created by the Q-SCP algorithm and allowing more requests to be admitted. In addition, the CLFirst algorithm achieves less system throughput than the DCFirst algorithm. The reason behind this is that the CLFirst algorithm first saturates the cloudlets before using instances in data centers. In other words, the CLFirst algorithm will keep assigning requests to cloudlets in the mobile edge cloud if there are still idle VNF instances and available computing resources. Recall that the objective is to maximize the accumulative data volume of admitted requests. The CLFirst algorithm first admits requests with high data volume. However, since the cloudlets have computing resource capacities, they may not be able to admit too many requests. The remaining requests that cannot be admitted by the cloudlets may also be rejected by the data centers due to the long transmission delay from APs to remote data centers. Yet the DCFirst algorithm has abundant resources to admit many requests with high delay requirements. Furthermore, we can see that the Heu algorithm has the highest operational cost since it admits the highest number of requests. However, the average cost for each request by the Heu algorithm is the lowest, as it prefers cloudlets or data centers that could save transmission cost, processing cost, or instantiation cost. As shown in Figure 7(d), the Heu algorithm incurs lower delay for each request than the DCFirstbut it has a higher delay than the CLFirst algorithm. The reason is that the CLFirst algorithm prefers cloudlets with a lower delay for each request, whereas the Heu algorithm sometimes chooses data centers to achieve a higher system throughput by sacrificing the delay.

7.3 Performance Evaluation of the Appro Algorithm

We then evaluate the performance of the Appro algorithm against that of the Q-SCP, CLFirst, and DCFirst algorithms, respectively, by varying the ratio of the number of cloudlets to the number of switches from 0.1 to 0.3. From Figure 8(a), it can be seen that the Appro algorithm delivers a higher system throughput than the other algorithms because the Appro algorithm considers the placement of value-added functions of each service chain, whereas the Heu algorithm treats each service function individually. In addition, the Appro algorithm concurrently deals with multiple requests, whereas the other mentioned algorithms can only deal with the requests one by one. Meanwhile, the system throughput delivered by the Q-SCP algorithm is slightly higher than that of the DCFirst algorithm because the Q-SCP algorithm consolidates VNFs into a single location and reduces the transmission delay of data packets, so more requests can meet their delay requirements. From Figure 8(b), we can see that the CLFirst algorithm has the lowest operational cost because its system throughput is also the lowest one. As can be seen from Figure 8(c), the average



(a) System throughput

(b) Total operational cost

(c) Average operational cost

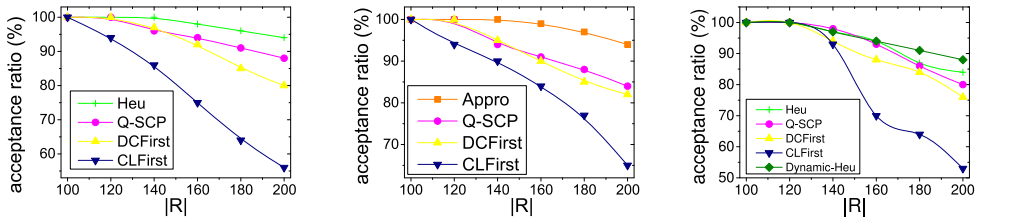
(d) Average delay per request

Fig. 9. The performance of algorithms Dynamic_Heu, Heu, Q-SCP, CLFirst and DCFirst in network AS1755.

operational cost of the CLFirst and Q-SCP algorithms are higher than that of the DCFirst algorithm because they both make use of more cloudlet resources and the average operational cost of the DCFirst algorithm is higher than that of the Appro algorithm because it uses more link resources.

7.4 Performance Evaluation of the Dynamic_Heu Algorithm

We now evaluate the performance of the Dynamic_Heu algorithm against that of the Heu, Q-SCP, CLFirst, and DCFirst algorithms, respectively, by varying the ratio of the number of cloudlets to the number of switches from 0.1 to 0.3. From Figure 9(a), we can see that the system throughput of the Dynamic_Heu algorithm is slightly higher than that of the Heu algorithm because when the IoT device moves, the packets of requests in the Heu algorithm are still transmitted through the original routing path, which causes a small number of requests to be rejected because of the violation on delay requirements. In addition, the packet rates of requests may increase and cause delay violations. The Dynamic_Heu algorithm reduces the likelihood of violating the delay requirements of requests by relocating the VNFs. We can see from Figure 9(b) and (c) that the total and average operational costs of the DCFirst algorithm increase significantly when the IoT device moves, and the DCFirst algorithm needs to use more link resources for the communication between the IoT device and the IoT applications in the remote cloud data centers. As can be seen from Figure 9(d), the average delays delivered by the Dynamic_Heu and Heu algorithms are similar because the Heu algorithm has a great possibility in selecting a low-latency routing path. The Heu algorithm implements a request via low-latency paths, whereas the Dynamic_Heu algorithm dynamically relocates VNFs to avoid delay violations. However, the average delay of the DCFirst algorithm is extremely higher than the others because it uses more link resources to transmit the data packets of the request when the IoT device of the request moves.



(a) Acceptance ratio of Heu (b) Acceptance ratio of Appro (c) Acceptance ratio of Dynamic_Heu

Fig. 10. The impact of the number of requests on the acceptance ratios of different algorithms.

7.5 Impact of Request Numbers on the Acceptance Ratios of Different Algorithms

We finally investigate the impact of network size on the acceptance ratio of the proposed algorithms by setting the network size to 100, the ratio of the number of cloudlets to the number of switches to 0.2, and varying the number of requests from 100 to 200. We can see from Figure 10(a) that the acceptance ratio of the Heu algorithm is the highest one, whereas the acceptance ratio of the CLFirst algorithm is the lowest one, and the Q-SCP algorithm is slightly better than algorithm DCFirst. We can see similar results in Figure 10(b) as well. However, as seen in Figure 10(a) and (c), when the IoT device moves, the acceptance ratios of the Heu and Q-SCP algorithms decrease slightly because of the violation of delay requirements. The CLFirst algorithm is hardly affected because of the low average delay, which satisfies the delay requirement of the request even if the IoT device moves. However, through comparison between Figure 10(c) and (d), we can see that when the volume of data packets changes, the acceptance ratio of the CLFirst algorithm drops significantly. This is because the combined effect of the IoT device movement and the changes on the volume of packets for requests leads to a significant increase on delays, which incurs the decline of the acceptance ratio. As can be seen in Figure 10(b) and (c), the acceptance ratio of the DCFirst algorithm has a significant drop because the movement of IoT devices increases the transmission delay, which violates delay requirements of requests.

8 CONCLUSION

In this article, we considered the VNF service chaining provisioning problem for IoT applications in a two-tier cloud network with both local edge clouds and remote distributed data centers. We first formulated a novel throughput maximization problem, with the objective of maximizing the system throughput while meeting the capacity constraints on cloudlets and remote data centers. We then proposed an efficient heuristic for the problem that makes a joint decision of placing VNFs into cloudlets and data centers, chaining the VNFs together, and finding a routing path for each request. We also devised an approximation algorithm with an approximation ratio for a special case of the problem without end-to-end delay requirements and special service chain requirements. Next, we considered the mobility and energy awareness of IoT applications by developing RL-based heuristics for dynamic VNF placements. We finally evaluated the performance of the proposed algorithms by simulations, and simulation results show that the performance of the proposed algorithms are promising.

ACKNOWLEDGMENTS

We would like to thank the three anonymous referees and the associate editor for their expertise comments and constructive suggestions, which have helped us improve the quality and presentation of the article greatly.

REFERENCES

[1] HPE Bladesystem Blade Servers. Retrieved April 30, 2020 from https://www.hpe.com/emea_europe/en/integrated-systems/bladesystem.html.

[2] Bernardetta Addis, Dallal Belabed, Mathieu Bouet, and Stefano Secci. 2015. Virtual network functions placement and routing optimization. In *Proceedings of the 2015 IEEE 4th International Conference on Cloud Networking (CloudNet'15)*. IEEE, Los Alamitos, CA, 171–177.

[3] Bilal R. Al-Kaseem and Hamed S. Al-Raweshidyhamed. 2017. SD-NFV as an energy efficient approach for M2M networks using cloud-based 6LoWPAN testbed. *IEEE Internet of Things Journal* 4, 5 (2017), 1787–1797.

[4] Gabriel Brown and H. Reading. 2015. *Service Chaining in Carrier Networks*. White Paper. Heavy Reading.

[5] Yang Chen and Jie Wu. 2018. NFV middlebox placement with balanced set-up cost and bandwidth consumption. In *Proceedings of the 47th International Conference on Parallel Processing*. ACM, New York, NY, 14.

[6] Rami Cohen, Liane Lewin-Eytan, Joseph Seffi Naor, and Danny Raz. 2014. On the effect of forwarding table size on SDN network utilization. In *Proceedings of the 2014 IEEE Conference on Computer Communications (IEEE INFOCOM'14)*. IEEE, Los Alamitos, CA, 1734–1742.

[7] Rami Cohen, Liane Lewin-Eytan, Joseph Seffi Naor, and Danny Raz. 2015. Near optimal placement of virtual network functions. In *Proceedings of the 2015 IEEE Conference on Computer Communications (IEEE INFOCOM'15)*. IEEE, Los Alamitos, CA, 1346–1354.

[8] Richard Cziva, Christos Anagnostopoulos, and Dimitrios P. Pezaros. 2018. Dynamic, latency-optimal VNF placement at the network edge. In *Proceedings of the 2018 IEEE Conference on Computer Communications (IEEE INFOCOM'18)*. IEEE, Los Alamitos, CA, 693–701.

[9] Richard Cziva and Dimitrios P. Pezaros. 2017. Container network functions: Bringing NFV to the network edge. *IEEE Communications Magazine* 55, 6 (2017), 24–31.

[10] Wanfu Ding, Wen Qi, Jianping Wang, and Biao Chen. 2015. OpenSCaaS: An open service chain as a service platform toward the integration of SDN and NFV. *IEEE Network* 29, 3 (2015), 30–35.

[11] Vajiheh Farhadi, Fidan Mehmeti, Ting He, Tom La Porta, Hana Khamfroush, Shiqiang Wang, and Kevin S. Chan. 2019. Service placement and request scheduling for data-intensive applications in edge clouds. In *Proceedings of the 2019 IEEE Conference on Computer Communications (IEEE INFOCOM'19)*. IEEE, Los Alamitos, CA, 1279–1287.

[12] Andrey Gushchin, Anwar Walid, and Ao Tang. 2015. Scalable routing in SDN-enabled networks with consolidated middleboxes. In *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. ACM, New York, NY, 55–60.

[13] Nicolas Herbaut, Daniel Negru, George Xilouris, and Yiping Chen. 2015. Migrating to a NFV-based home gateway: Introducing a surrogate vnf approach. In *Proceedings of the 2015 6th International Conference on the Network of the Future (NOF'15)*. IEEE, Los Alamitos, CA, 1–7.

[14] Huawei Huang, Song Guo, Jinsong Wu, and Jie Li. 2017. Service chaining for hybrid network function. *IEEE Transactions on Cloud Computing* 7, 4 (2017), 1082–1094.

[15] Huawei Huang, Peng Li, and Song Guo. 2017. Traffic scheduling for deep packet inspection in software-defined networks. *Concurrency and Computation: Practice and Experience* 29, 16 (2017), e3967.

[16] Meitian Huang, Weifa Liang, Zichuan Xu, Wenzheng Xu, Song Guo, and Yinlong Xu. 2016. Dynamic routing for network throughput maximization in software-defined networks. In *Proceedings of the 35th Annual IEEE International Conference on Computer Communications(IEEE INFOCOM'16)*. IEEE, Los Alamitos, CA, 1–9.

[17] Mike Jia, Jiannong Cao, and Weifa Liang. 2015. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing* 5, 4 (2015), 725–737.

[18] Mike Jia, Weifa Liang, and Zichuan Xu. 2017. QoS-aware task offloading in distributed cloudlets with virtual network function services. In *Proceedings of the 20th ACM International Conference on Modelling, Analysis, and Simulation of Wireless and Mobile Systems*. ACM, New York, NY, 109–116.

[19] Mike Jia, Weifa Liang, Zichuan Xu, and Meitian Huang. 2016. Cloudlet load balancing in wireless metropolitan area networks. In *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM'16)*. IEEE, Los Alamitos, CA, 1–9.

[20] Stavros G. Kolliopoulos and Clifford Stein. 2001. Approximation algorithms for single-source unsplittable flow. *SIAM Journal on Computing* 31, 3 (2001), 919–946.

[21] Tung-Wei Kuo, Bang-Heng Liou, Kate Ching-Ju Lin, and Ming-Jer Tsai. 2018. Deploying chains of virtual network functions: On the relation between link and server usage. *IEEE/ACM Transactions on Networking* 26, 4 (2018), 1562–1576.

[22] Yang Li, Linh Thi Xuan Phan, and Boon Thau Loo. 2016. Network functions virtualization with soft real-time guarantees. In *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM'16)*. IEEE, Los Alamitos, CA, 1–9.

[23] Tamás Lukovszki and Stefan Schmid. 2015. Online admission control and embedding of service chains. In *Proceedings of the International Colloquium on Structural Information and Communication Complexity*. 104–118.

[24] Yu Ma, Weifa Liang, Zichuan Xu, and Song Guo. 2018. Profit maximization for admitting requests with network function services in distributed clouds. *IEEE Transactions on Parallel and Distributed Systems* 30, 5 (2018), 1143–1157.

[25] Joao Martins, Mohamed Ahmed, Costin Raiciu, Vladimir Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici. 2014. ClickOS and the art of network function virtualization. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI'14)*. 459–473.

[26] Hendrik Moens and Filip De Turck. 2014. VNF-P: A model for efficient placement of virtualized network functions. In *Proceedings of the 10th International Conference on Network and Service Management (CNSM'14) and Workshop*. IEEE, Los Alamitos, CA, 418–423.

[27] Carla Mouradian, Narjes Tahghigh Jahromi, and Roch H. Glitho. 2018. NFV and SDN-based distributed IoT gateway for large-scale disaster management. *IEEE Internet of Things Journal* 5, 5 (2018), 4119–4131.

[28] Yeonghun Nam, Sooeun Song, and Jong-Moon Chung. 2016. Clustered NFV service chaining optimization in mobile edge clouds. *IEEE Communications Letters* 21, 2 (2016), 350–353.

[29] Jan Plachy, Zdenek Becvar, and Emilio Calvanese Strinati. 2016. Dynamic resource allocation exploiting mobility prediction in mobile edge computing. In *Proceedings of the 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC'16)*. IEEE, Los Alamitos, CA, 1–6.

[30] Yaozhong Song, Stephen S. Yau, Ruozhou Yu, Xiang Zhang, and Guoliang Xue. 2017. An approach to QoS-based task distribution in edge computing networks for IoT applications. In *Proceedings of the 2017 IEEE International Conference on Edge Computing (EDGE'17)*. IEEE, Los Alamitos, CA, 32–39.

[31] Neil Spring, Ratul Mahajan, and David Wetherall. 2002. Measuring ISP topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review* 32 (2002), 133–145.

[32] Yuxuan Sun, Sheng Zhou, and Jie Xu. 2017. EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks. *IEEE Journal on Selected Areas in Communications* 35, 11 (2017), 2637–2646.

[33] Megan Thomas and Ellen W. Zegura. 1994. *Generation and Analysis of Random Graphs to Model Internetworks*. Technical Report. College of Computing, Georgia Institute of Technology, Atlanta, GA.

[34] Petra Vizarreta, Massimo Condoluci, Carmen Mas Machuca, Toktam Mahmoodi, and Wolfgang Kellerer. 2017. QoS-driven function placement reducing expenditures in NFV deployments. In *Proceedings of the 2017 IEEE International Conference on Communications (ICC'17)*. IEEE, Los Alamitos, CA, 1–7.

[35] Qiufen Xia, Weifa Liang, and Wenzheng Xu. 2013. Throughput maximization for online request admissions in mobile cloudlets. In *Proceedings of the 38th Annual IEEE Conference on Local Computer Networks*. IEEE, Los Alamitos, CA, 589–596.

[36] Qiufen Xia, Weifa Liang, Zichuan Xu, and Bingbing Zhou. 2014. Online algorithms for location-aware task offloading in two-tiered mobile cloud environments. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. IEEE, Los Alamitos, CA, 109–116.

[37] Jie Xu, Lixing Chen, and Pan Zhou. 2018. Joint service caching and task offloading for mobile edge computing in dense networks. In *Proceedings of the 2018 IEEE Conference on Computer Communications (IEEE INFOCOM'18)*. IEEE, Los Alamitos, CA, 207–215.

[38] Zichuan Xu, Weifa Liang, Meitian Huang, Mike Jia, Song Guo, and Alex Galis. 2019. Efficient NFV-enabled multicasting in SDNs. *IEEE Transactions on Communications* 67, 3 (2019), 2052–2070.

[39] Zichuan Xu, Weifa Liang, Mike Jia, Meitian Huang, and Guoqiang Mao. 2018. Task offloading with network function requirements in a mobile edge-cloud network. *IEEE Transactions on Mobile Computing* 18, 11 (2018), 2672–2685.

[40] Binxu Yang, Wei Koong Chai, George Pavlou, and Konstantinos V. Katsaros. 2016. Seamless support of low latency mobile applications with NFV-enabled mobile edge-cloud. In *Proceedings of the 2016 5th IEEE International Conference on Cloud Networking (CloudNet'16)*. IEEE, Los Alamitos, CA, 136–141.

[41] Binxu Yang, Wei Koong Chai, Zichuan Xu, Konstantinos V. Katsaros, and George Pavlou. 2018. Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications. *IEEE Transactions on Network and Service Management* 15, 1 (2018), 475–488.

[42] Ruozhou Yu, Guoliang Xue, and Xiang Zhang. 2018. Application provisioning in fog computing-enabled Internet-of-Things: A network perspective. In *Proceedings of the 2018 IEEE Conference on Computer Communications (IEEE INFOCOM'18)*. IEEE, Los Alamitos, CA, 783–791.

[43] Qixia Zhang, Fangming Liu, and Chaobing Zeng. 2019. Adaptive interference-aware VNF placement for service-customized 5G network slices. In *Proceedings of the 2019 IEEE Conference on Computer Communications (IEEE INFOCOM'19)*. IEEE, Los Alamitos, CA, 2449–2457.

Received April 2019; revised November 2019; accepted March 2020