Task Offloading with Network Function Requirements in a Mobile Edge-Cloud Network

Zichuan Xu, *Member, IEEE*, Weifa Liang, *Senior Member, IEEE*, Mike Jia, Meitian Huang, and Guoqiang Mao, *Fellow, IEEE*

Abstract—Pushing the cloud frontier to the network edge close to mobile users has attracted tremendous interest not only from cloud operators but also from network service providers. In particular, the deployment of *cloudlets* in metropolitan area networks enables network service providers to provide low-latency services to mobile users through implementing their specified virtualized network functions (VNFs) while meeting their Quality-of-Service (QoS) requirements. In this paper, we formulate a novel task offloading problem in a mobile edge-cloud network, where each offloading task requests a specified network function with a tolerable delay. We aim to maximize the number of requests admitted while minimizing the operational cost of admitted requests within a finite time horizon, through either sharing existing VNF instances or creating new VNF instances in cloudlets. We first show that the problem is NP-hard, and then devise an efficient online algorithm for the problem by reducing it to a series of minimum weight maximum matching problems. Considering dynamic changes of task offloading request patterns over time, we further develop an effective prediction mechanism for new VNF instance creations and idle VNF instance releases to further lower the operational cost of the network service provider. Also, we devise an online algorithm with a competitive ratio for a special case of the problem where the delay requirements of requests are negligible. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results indicate that the proposed algorithms are promising.

Index Terms—Mobile edge-cloud networks, task offloading, operational cost minimization, network function virtualization, throughput maximization, online algorithms, resource allocations in cloudlets

1 Introduction

Mobile devices such as smart phones and tablets are becoming the main communication tools. Meanwhile, face recognition, natural language processing, interactive gaming, and augmented reality are emerging as new mobile applications. Such mobile applications request a large amount of computing resource for performance enhancement, thereby leading to high-level energy consumptions. They also require various network function services including firewalls, intrusion detection systems, and load balancers, to guarantee correct and secure execution of the applications. Most of such applications have stringent Quality-of-Service (QoS) requirements (i.e., extremely low service response delays). However, due to the portable size of mobile devices, their computing, storage, and battery capacities are very limited, many computing- and/or storage-intensive applications may not be suitable to run in mobile devices, thereby restricting the capability of mobile devices. One promising technique to leverage the capability of mobile devices is to offload their tasks to nearby cloudlets in a mobile edge-cloud network via WiFi or 5G for processing.

- *Z. Xu is with the School of Software, Dalian University of Technology, Dalian 116020, China. E-mail: z.xu@dlut.edu.cn.*
- *W. Liang, M. Jia, and M. Huang are with the Research School of Computer Science, The Australian National University, Canberra, ACT 0200, Australia. E-mail: wliang@cs.anu.edu.au, {u5515287, meitian.huang}@anu.edu.au.*
- *G. Mao is with the School of Computing and Communications, University of Technology Sydney, Sydney, NSW 2007, Australia. E-mail: Guoqiang.Mao@uts.edu.au.*

Manuscript received 3 May 2018; revised 5 Oct. 2018; accepted 19 Oct. 2018. Date of publication 23 Oct. 2018; date of current version 30 Sept. 2019. (Corresponding author: Weifa Liang.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2018.2877623

To meet ever-growing resource demands of offloading tasks from mobile users with stringent QoS requirements, network service providers usually instantiate some frequently demanded VNF instances of network functions at cloudlets in mobile edge-cloud networks. The instantiation of VNF instances at the edge of a network can shorten the access latency of network services and save time in creating new VNF instances. Provisioning network services with different types of VNFs in a mobile edge-cloud network poses many challenges. For example, how many VNF instances are needed to be instantiated such that the computing resource of cloudlets is fully utilized while the cost and delay of their instantiations are minimized? How should offloaded tasks be assigned to different cloudlets while meeting their QoS requirements and minimizing the admission cost by utilizing existing VNF instances they requested? Finally, can the demanded number of VNF instances at each cloudlet be predicted? In this paper we will address the aforementioned challenges.

To the best of our knowledge, this paper is the first one to explore the sharing of existing VNF instances of network functions in cloudlets for cost-efficient task offloading while meeting QoS requirements of individual offloaded tasks. We formulate a novel QoS-aware task offloading optimization problem and developing efficient solutions to the problem. Due to the nature of dynamic changes of offloading request patterns over time, we develop an effective prediction mechanism to predict VNF instance demands at each cloudlet to further reduce the operational cost of the service provider, through dynamic creations and releases of VNF instances at cloudlets in the network.

The main contributions of this paper are as follows. We first formulate a novel QoS-aware task offloading problem

in a mobile edge-cloud network that consists of a number of cloudlets co-located with Access Points (APs). We aim to maximize the number of admissions of offloading requests within a finite time horizon while minimizing the admission cost of admitted requests, assuming that different task offloading requests request different VNF services and have different end-to-end delay requirements. We then devise an efficient algorithm for admissions of a set of task offloading requests through a non-trivial reduction that reduces the problem to a series of minimum weight maximum matching problems. We also develop an efficient online algorithm for dynamic offloading request admissions, in addition to an effective prediction mechanism to predict the demanded number of VNF instances of each different network function at each cloudlet. For a special case of the problem where the end-to-end delay requirement is negligible, we devise an online algorithm with a provable competitive ratio, by adopting the primal-dual dynamic updating method [5]. We finally evaluate the performance of the proposed algorithms through experimental simulations, and results demonstrate that the proposed algorithms are promising.

The remainder of the paper is arranged as follows. Section 2 will survey the state-of-the-art on task offloading in mobile edge computing environments, and distinguish our work in this paper from previous studies. Section 3 will introduce the system model, notations and problem definitions. Section 4 will develop a prediction mechanism, and devise algorithms for the problem based on the built prediction mechanism. Section 5 will develop an online algorithm with a provable competitive ratio for a special case of the problem. Section 6 will provide some experimental results on the performance of the proposed algorithms, and Section 7 concludes the paper.

2 Related Work

As a key enabling technology of 5G, mobile edge-cloud networks have gained tremendous attention recently [15]. Also, with the emergence of complicated and resource-hungry mobile applications, offloading user tasks to cloudlets of a nearby mobile edge-cloud network is becoming an important approach to reduce the energy consumption of mobile devices and improve mobile user QoSs.

Extensive studies on task offloading in mobile edge-cloud networks have been conducted [1], [2], [3], [4], [9], [11], [15], [19], [20], [23]. For example, Chen et al. [1] investigated the problem of task offloading for mobile edge computing in a software-defined ultra-dense network with the aim to minimize the total execution duration of all tasks. However, they assumed that edge clouds are deployed at each base station, and VNF placement is not considered. Chen et al. [2] investigated the problem of multi-user computation offloading for mobile edge clouds in a multi-channel wireless interference environment. They formulated their problem as a multi-player game and designed a mechanism for the problem as a Nash equilibrium. However, they did not consider the placement of virtualized network functions. Zhang [24] treated data offloading as a coalitional game-based pricing scheme, formulated the problem as coalitional non-transferable utility game, and conducted theoretical and empirical analysis on their proposed solution. Song et al. [18] investigated the task assignment problem in a mobile edge network with node, link, and security constraints, and proposed a heuristic for it. Jia et al. [11] dealt with minimizing the maximum delay among offloaded tasks in a distributed cloudlet network through balancing the workload among cloudlets. Xu et al. [21], [22] devised efficient approximation and online algorithms to the assignment of offloading requests to cloudlets. Xia et al. [19] considered opportunistic task offloading under link bandwidth, mobile device energy, and cloudlet computing capacity constraints.

None of the mentioned studies considered task offloading with VNF service requirement. However, there are several dedicated studies focussing on VNF placement in conventional networks or mobile edge network [13], [16], [23]. For example, Kuo et al. [13] studied the problem of service chain embedding in SDNs with both node and edge capacity constraints. They proposed a primal-dual based solution with bounded error, and a randomized approximation algorithm with average performance guarantees. All the aforementioned studies assumed that each offloaded task will be assigned an amount of dedicated computing resource. No consideration has ever been given for existing VM or VNF instance sharing in cloudlets, not to mention there is any prediction mechanism to predict future demands on VNF instances of different network functions by their creations and releases to further reduce the service costs and delays. With imminent 5G technology, provisioning extremely low-latency services has emerged as a new trend, and existing studies ignore the latency in either wireless access networks (from mobile devices to APs) or mobile edge networks (between APs and the core network). It must be mentioned that the work in this paper is an extension of the work from our conference paper [10].

3 Preliminaries

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

3.1 System Model

Given a mobile edge-cloud network $G = (V \cup C, E)$ where V is the set of AP nodes, C is a set of cloudlets co-located with some AP nodes, and E is the set of links between AP nodes. Since cloudlets are usually co-located with APs in coffee shops, base stations, libraries, school buildings, airports, or shopping malls in metropolitan areas within the proximity of mobile users, only a limited number of servers can be installed within each cloudlet, due to the space or cooling limitation of the locations. Thus, each cloudlet $cl_j \in C$ is assumed to have computing capacity cap_j with $1 \leq j \leq |C|$, where $|C|$ is the cardinality of set C. Assuming that time is divided into equal time slots, the amounts of available computing and storage resources in each cloudlet at each time slot vary, due to admissions and departures of offloading task requests. Let $cap_j(t)$ be the amount of available computing resource of cloudlet cl_j at time slot t with $1 \leq t \leq T$ and $1 \leq j \leq |C|$, where T is the monitoring period in terms of numbers of time slots. Fig. 1 is an example of a mobile edge-cloud network.

3.2 Task Offloading and VNF Instances

With the development of mobile technology, mobile applications are becoming more and more complicated, by consuming not only energy, computing and storage resource but also requesting advanced network services that are represented as various VNFs. Offloading tasks from mobile devices can save both energy and computing resources of mobile devices while meeting increasing demands on

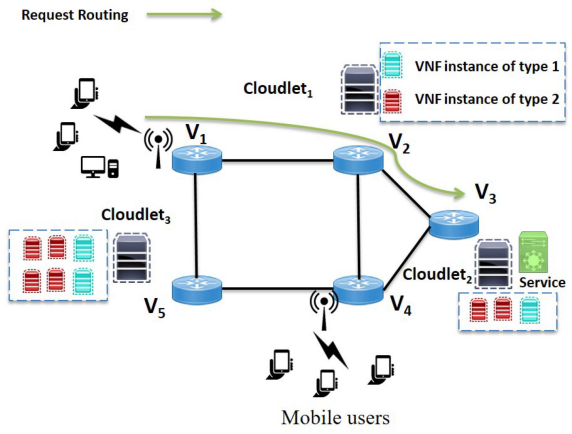


Fig. 1. An example of a mobile edge-cloud network.

advanced network services of the requests. To this end, computing resource in each cloudlet is used to instantiate a certain number of VNFs running in VMs, referred to as *VNF instances*. We thus assume that there is a set $\mathcal{F}$ of VNFs in the mobile edge-cloud network G. Denote by f_i a type-i of VNF in $\mathcal{F}$ with $1 \leq i \leq |\mathcal{F}|$.

An implementation of a type-i network function f_i in a VM in a cloudlet cl_j is termed as an instance of $f_i \in \mathcal{F}$. Without loss of generality, we assume that the amount of computing resource in cl_j allocated to a VNF instance of f_i is to guarantee its maximal packet processing rate μ_{ij} in cl_j. Denote by RC^{unit} the amount of resource that is needed to process a unit packet rate, and the amount of resources needed by an instance of f_i is $\mu_{ij} \cdot RC^{unit}$. There may and may not have multiple VNF instances of each $f_i \in \mathcal{F}$ in a cloudlet. We further assume that each cloudlet has instantiated some VNF instances of each f_i already. Denote by $n_{ij}(t)$ the number of existing VNF instances of network function f_i in cloudlet cl_j at time slot t. Each VNF instance can process one or multiple user requests that specify this type of VNF and the aggregate packet rate in the VNF instance is no more than its maximum processing rate μ_{ij}.

Each mobile device can offload its task to some instances of VNF f_i it requested via a nearby AP. Let $S(t)$ be the set of offloading task requests at time slot t. Each user task request $r_k \in S(t)$ is represented by a tuple $(v_k, F_k, \lambda_k, \tau_k, d_k)$, where v_k is of the user's closest AP $v_k \in V$, F_k is the network function that r_k requests, λ_k is the packet rate of request r_k, τ_k is the duration of request r_k, and d_k is the end-to-end delay requirement of the request. Note that the value of λ_k can be derived from historical information of similar types of user requests. We thus assume that the packet rate of each request is given as a priori.

3.3 End-to-End Delays of Offloading Requests

The end-to-end delay experienced by each admitted request r_k includes *the upload delay* of uploading the packet traffic of a mobile user to its nearby AP, *the queuing delay* in the cloudlet cl_j for the processing of the packet traffic on the requested VNF instance, *the packet processing delay* in the VNF instance, *the instantiation delay* of creating a VNF instance if needed, *the network latency* between the AP and a cloudlet cl_j, and *the result return delay* from the AP to the user. For the sake of convenience, we assume that the packet upload delay and the result return delay are equal. These delays are defined as follows.

Upload Delay. Packets of request r_k are uploaded to its nearby AP of its mobile user. Assuming that λ_k is the packet rate of r_k and the bandwidth of the AP is B_k, the achieved data rate W_k (bits per second) via the wireless channel of the AP for r_k is

$$W_k = B_k \log_2\left(1 + \frac{p_k \cdot h_k}{\sigma^2 + I_k}\right), \tag{1}$$

where σ^2 is the noise power of mobile devices and I_k is the inter-cell interference power [1], h_k is the channel gain between the user of request r_k and the AP, and p_k is the transmission power of the user with request r_k.

Let sp be a packet size in bits. The upload delay of r_k is

$$d(k, \lambda_k) = \lambda_k \cdot sp / W_k. \tag{2}$$

Notice that when $sp \cdot W_k < \lambda_k$, the packet rate of request r_k can be decreased due to the incurred delay of transmitting packets. Such decreased packet rates may affect the arrival rates of the queues in each cloudlet and their queuing delays. For the sake of simplicity, we assume that there is a buffer at each AP and a buffering mechanism is available to allow the decreased packet rate to reach its original rate. Notice that the impact of the queue in each AP on the arrival packet rate of the queue in each cloudlet depends on many factors such as the buffer size at each AP and the adopted dispatch methods. This is out the scope of this paper, we thus consider it as our future work.

Queuing Delay and Processing Delay. Each packet of r_k with packet rate λ_k will be queued in the cloudlet assigned to the request for a VNF instance of network function F_k prior to its processing by the instance, which will incur both queuing delay in the cloudlet and the processing delay in the VNF instance. Since different requests need different types of network functions, we assume that there is an $M/M/n$ queue at each cloudlet for each type of network function $f_i \in \mathcal{F}$. The requests in each queue will be processed by its type of VNF instances. The average queuing delay d_{ij}^{queue} of the $M/M/n$ queue for network function f_i at cloudlet cl_j is

$$d_{ij}^{queue}(\Lambda_{ij}) = 1/(n_{ij}(t) \cdot \mu_{ij} - \Lambda_{ij}), \tag{3}$$

where Λ_{ij} is the sum of packet rates of all requests that request the VNF instances of f_i in cloudlet cl_j, i.e.,

$$\Lambda_{ij} = \sum_{r_{j'} \in R_{ij}} \lambda_{j'} + \lambda_k, \tag{4}$$

The processing delay p_{ij} of a packet in a VNF instance of f_i of cloudlet cl_j is

$$p_{ij} = 1/\mu_{ij}. \tag{5}$$

Instantiation Delay. Without loss of generality, we assume that the instantiation delay of a VNF instance in a cloudlet is a given constant d_i^{ins} for a VNF instance of f_i.

Network Latency. Assuming that the routing path for the data traffic in G between the user of a request and the cloudlet for processing the user's data traffic is a shortest path between them, the network latency of a request r_k thus is the accumulative delay incurred in the links of the shortest path p_{v_k,cl_j} between its closest AP v_k and its assigned cloudlet cl_j. As AP nodes in G are connected by high-speed optical cables, there is sufficient bandwidth with each such a link for data traffic, the queuing delay in each link thus is negligible. Let $d(e)$ be the delay on link e in G. The network

latency d_{kj}^{met} experienced by routing data traffic of r_k using a shortest path p_{v_k,cl_j} can be defined as

$$d_{kj}^{met} = \sum_{e \in p_{v_k,cl_j}} d(e). \quad (6)$$

The end-to-end delay D_k experienced by implementing request r_k in a VNF instance of F_k $(= f_i)$ at cloudlet cl_j also depends on whether the VNF instance is existent or newly created. If the VNF instance is existent, there will be no instantiation delay for the VNF instance. Otherwise, a new VNF instance needs to be instantiated, and there will be no queuing delay in cloudlet cl_j incurred since the request implementation is the very first one on it.

Having the equations Eqs. (1) to (6), the value of D_k is

$$D_k = \begin{cases} 2d(k,\lambda_k) + d_{ij}^{queue}(\Lambda_{ij}) + p_{ij} + 2d_{kj}^{met} & \text{if } n_{ij}(t) > 0, \\ 2d(k,\lambda_k) + d_i^{ins} + p_{ij} + 2d_{kj}^{met} & \text{otherwise,} \end{cases} \quad (7)$$

where $F_k = f_i$. Request r_k is *admissible* if there is an admission schedule in which D_k is no greater than d_k, i.e.,

$$D_k \leq d_k. \quad (8)$$

3.4 Admission Cost

Implementing each request needs to guarantee that its traffic is processed by VNFs and transferred to its destination, which consumes both computing resource in cloudlets and network bandwidth resource in links of the edge-cloud network G, thereby incurring the cost of resource consumptions. We refer to this cost as *the admission cost* of each request in the rest of the paper.

Given a request $r_k \in S(t)$ with a specified network function F_k $(= f_i)$, its implementation can either make use of an existing VNF instance of F_k $(= f_i)$ in cloudlet cl_j if it joins in other admitted requests or create a new VNF instance. Specifically, let R_{ij} be the set of offloaded requests with network function f_i in cloudlet cl_j when r_k is being considered for admission. If the admission of r_k to R_{ij} will not violate the delay constraint of any admitted request in $R_{ij} \cup \{r_k\}$, the cost by admitting r_k in cloudlet cl_j then is the cost sum of its data packet transmission cost (between its location via its nearby AP) and cloudlet cl_j and its processing cost at cl_j. Otherwise, we allocate the demanded resources for r_k by creating more VNF instances for F_k $(= f_i)$ in cl_j if there are available computing resources in cloudlet cl_j. Since VNF instance creation takes time and consumes a certain amount of computing resource, we assume that the creation of new instances incurs a cost, and different VNF instance creations result in different costs.

The admission cost $cost(k)$ per packet of the data traffic of r_k thus is the sum of the packet routing cost, the creation of new instances for r_k, and the processing cost in a VNF instance of f_i $(= F_k)$ assuming that the VNF instance is deployed in cloudlet cl_j, i.e.,

$$cost(k) = \begin{cases} c_i^{ins} + \lambda_k \tau_k (C(f_i, cl_j) + \sum_e \in P_{v_k,cl_j} c(e)) \\ \qquad \text{if a type}-i \text{ VNF instance is created for } r_k \\ \lambda_k \tau_k (C(f_i, cl_j) + \sum_e \in P_{v_k,cl_j} c(e)) \qquad \text{otherwise,} \end{cases} \quad (9)$$

where P_{v_k,cl_j} is the shortest path between the request user location v_k and cloudlet cl_j to process the request VNF, $c(e)$ is the cost of transmitting a packet via edge e, $C(f_i, cl_j)$ is the processing cost of a packet in an instance of f_i in cl_j, and c_i^{ins} is the cost of instantiating a type-i VNF instance.

3.5 Problem Definition

Given a mobile edge-cloud network $G(V \cup C, E)$, a set $\mathcal{F}$ of network functions provided by the network, a finite time horizon T, a set of task offloading requests $S(t)$ at a time slot t in which each request r_k has a packet rate λ_k and an end-to-end delay requirement d_k, assuming that some VNF instances of a network function $f \in \mathcal{F}$ have been instantiated in cloudlets of C, *the throughput maximization problem* in G is to find a schedule of request admissions during time horizon T such that as many as requests are admitted while the accumulative operational (admission) cost of the admitted requests is minimized, subject to the computing resource capacity on each cloudlet in C.

In other words, let $R(t)$ and $S(t)$ be the sets of admitted requests by an algorithm and all arrived requests at time slot t, respectively. Clearly, $R(t) \subseteq S(t)$ with $1 \leq t \leq T$, the throughput maximization problem is to maximize $\sum_{t=1}^{T} |R(t)|$ while minimizing $\sum_{t=1}^{T} \sum_{r_k \in R(t)} cost(k)$, subject to computing resource capacity on each cloudlet.

Theorem 1. *The throughput maximization problem in $G(V \cup C, E)$ is NP-hard.*

Proof. We consider a special case of the problem where there are only two cloudlets with identical computational capacities in the network and the monitoring period is only one time slot t. We assume that each request in $S(t)$ requests a different network function service without the end-to-end delay requirement. Our task is to assign the requests in $S(t)$ to the two cloudlets to see whether all the requests can be admitted. Clearly, for each request $r_k \in S(t)$, we need to create a VNF instance to implement its network function that demands $c(F_k)$ computing resources, subject to the computing capacities on the two cloudlets. We reduce the summation problem to this special throughput maximization problem as follows.

Given n positive integers $a_1, a_2, \ldots, a_n$, the summation problem is to partition the n integers into two subsets such that the sum of integers in each subset is equal, which is NP-hard. As the special case of the minimum operational cost problem is equivalent to the summation problem if we only consider the computing resource demands of the requests while ignoring the other costs, the throughput maximization problem thus is NP-hard, too. □

4 Online Algorithm for the Throughput Maximization Problem

In this section, we first consider request admissions in the beginning of each time slot t. We then propose an efficient online algorithm for the throughput maximization problem.

4.1 Algorithm for Task Offloading at Time Slot t

Given a set of requests $S(t)$ in the beginning of each time slot t, we aim to admit as many as requests in $S(t)$ while minimizing the accumulative operational cost and meeting their end-to-end delay requirements. The basic idea behind the proposed algorithm is to reduce the throughput maximization problem in G to a series of minimum weight maximum matching problems on a set of auxiliary bipartite graphs. The matched edges in the maximum matching in an auxiliary bipartite graph correspond to an assignment of task offloading requests to different cloudlets while meeting

the end-to-end delay requirement of each admitted request. The detailed reduction is as follows.

For each cloudlet cl_j, we construct a bipartite graph $G_j(t) = (X_j \cup \{x_{0,j}\} \cup Y_j, E_j; w)$. The nodes in $G_j(t)$ is partitioned into two disjoint sets. The first one is Y_j representing the set of requests that need to be assigned to existing or new instances of cloudlets. The second one is $X_j \cup \{x_{0,j} \mid 1 \leq j \leq |C|\}$, where X_j is the set of VNF instances in cloudlet cl_j with each VNF instance being represented by x_{ij} and node $x_{0,j}$ represents available computing resource for creating new instances for any VNF in cl_j.

The edges between nodes in Y_j and the ones in $X_j \cup \{x_{0,j} \mid 1 \leq j \leq |C|\}$ in E_j can be constructed as follows.

Let R_{ij} be the admitted requests that are using the VNF instance x_{ij}. There is an edge $(r_k, x_{ij}) \in E_j$ between request node $r_k \in Y_j$ and node $x_{ij} \in X_j$ if the following conditions are met: (i) f_i is the VNF of request r_k; and (ii) the addition of r_k into the set R_{ij} of admitted requests sharing x_{ij} does not violate the delay constraints of other admitted requests in R_{ij}. Notice that the addition of r_k into R_{ij} can impact the queuing delay of the executing requests in R_{ij}. To check its addition does not violate any delay requirements, we only need to recalculate the queuing delay by Eq. (3), assuming that r_k is admitted to R_{ij}. If neither of the two conditions is met, a new VNF instance for request r_k will be created in cl_j if cloudlet cl_j has sufficient computing resource. Then, an edge between r_k and $x_{0,j}$ is added to E_j if its demanded computing resource is no greater than $cap_j(t)$ and the end-to-end delay (including the instantiation delay) is no greater than d_k. The weight w_{kj} assigned to each edge $(r_k, x_{ij}) \in E_j$ is the admission cost of request r_k in cloudlet cl_j for the duration τ_k, as defined in Eq. (9).

An auxiliary bipartite graph $G(t) = \bigcup_{j=1}^{|C|} G_j(t) = (X(t) \cup Y(t), E(t); w)$ is then derived from $G_j(t)$ with $1 \leq j \leq |C|$, where $X(t) = \cup_{j=1}^{|C|}(X_j \cup \{x_{0,j}\})$, $Y(t) = \cup_{j=1}^{|C|} Y_j = \{r_1, r_2, \ldots, r_n\}$, and $E(t) = \cup_{j=1}^{|C|} E_j$.

To admit requests in $S(t)$ in the beginning of time slot t, the proposed algorithm proceeds iteratively. Let $G_1(t) = G(t)$. Within iteration l, $1 \leq l \leq |C|$, a minimum weight maximum matching $M_{t,l}$ in $G_l(t)$ is found. The demanded resources of the matched requests in $M_{t,l}$ are allocated, and these requests are then removed from $S(t)$. The available VNF instances and resources at each cloudlet are updated accordingly, and the next auxiliary bipartite graph $G_{l+1}(t)$ is constructed. This procedure continues until there does not have any matching in $G_{l+1}(t)$ or $l = |C| + 1$.

The union of all found minimum weight maximum matchings $\cup_{l=1}^{|C|} M_{t,l}$ forms a solution to the problem, i.e., each matched edge corresponds to an admission of a request in $S(t)$. The weighted sum $\sum_{l=1}^{L} c(M_{t,l})$ of all edges in $\cup_{l=1}^{L} M_{t,l}$ is the accumulative admission cost of admitted requests in $S(t)$, where L is the number of iterations which depends on the requests in $S(t)$ and $L \leq |C|$. Alternatively, $L \leq \max_{1 \leq l \leq L}\{deg(G_l(t))\}$ which $deg(G_l(t))$ is the maximum degree of nodes in auxiliary graph $G_l(t)$. The detailed algorithm is given in `Algorithm 1`, which is referred to as `OL_STS` in the rest of this paper.

4.2 Online Algorithm

So far we considered the admissions of task offloading requests at a single time slot t. In reality, request arrives and departs dynamically without the knowledge of future request arrivals. The request admissions at the current time slot impact the admissions of future requests due to resource occupations by them. In the following we deal with dynamic request admissions for a given time horizon T. We note that the VNF instances created for previous request admissions are still in the system despite that the admitted requests finished. These existing VNF instances will become idle if they are utilized by later requests, and their maintenance will be overwhelming. Meanwhile, newly arrived requests that demand other types of VNF instances cannot be met due to lack of sufficient resources. We here propose an effective prediction mechanism to determine various VNF instance creations and releases for cost savings.

Algorithm 1. OL_STS($G(t)$)

Input: $G(t)$, $|C|$ cloudlets with each having its available resource capacity $cap_j(t)$, the number of instances n_{ij} of VNFs of each $f_i \in \mathcal{F}$ in cloudlet cl_j, and a set of requests $S(t)$ at each time slot t with $1 \leq j \leq |C|$.

Output: maximize the number of requests admitted (i.e., a subset $S'(t) \subseteq S(t)$) for each time slot t while minimizing the total admission cost.

1: $M_t \leftarrow \emptyset$; $cost(t) \leftarrow 0$; /* the assignment of requests in $S(t)$while minimizing their implementation cost $cost(t)$ */
2: Construct the weighted bipartite graph $G(t)$;
3: $G_1(t) \leftarrow G(t)$; $l \leftarrow 1$;
4: **while** there is a maximum matching $M_{t,l}$ in $G_l(t)$ **do**
5: Find the minimum weight maximum matching $M_{t,l}$ in $G_l(t)$;
6: **if** $M_{t,l} \neq \emptyset$ **then**
7: $M_t \leftarrow M_t \cup M_{t,l}$, $c(M_{t,l}) \leftarrow \sum_{e \in M_{t,l}} w(e)$, $cost(t) \leftarrow cost(t) + c(M_{t,l})$;
8: Allocate resources to the matched requests in $M_{t,l}$;
9: Update the amounts of available resources and VNF instances of each network function in each cloudlet;
10: $S(t) \leftarrow S(t) \setminus r(M_{t,l})$; /* Remove requests in $M_{t,l}$ from $S(t)$, where $r(M_{t,l})$ is the set of matched requests in $M_{t,l}$ */
11: $l \leftarrow l + 1$;
12: Construct $G_l(t)$;
13: **return** M_t corresponds to the assignment of requests in $S(t)$, while $cost(t)$ is their admission cost.

To respond to changing request patterns over time, the system will perform resource collection through releasing the occupied resources by idle VNF instances back to the system if the maintenance overhead on the idle VNFs is beyond a given cost threshold after a certain number of time slots. Specifically, let $n_{ij}(t)$ be the number of VNF instances of network function f_i in cloudlet cl_j at time slot t and the actual usage number of the VNF instances of f_i be $n'_{ij}(t)$ ($\leq n_{ij}(t)$), the number of idle VNF instances of f_i in cloudlet cl_j at time slot t then become

$$\psi_{ij}(t) = n_{ij}(t) - n'_{ij}(t). \tag{10}$$

Let ϕ_{ij} be a fixed cost for the maintenance of an idle VNF instance of f_i in cloudlet cl_j per time slot. There is a given threshold θ ($\geq n_0 \cdot \max_{f_i \in \mathcal{F}}\{C(f_i)\}$) for the overhead of maintaining idle VNF instances in cloudlets. The system will release the resources occupied by the idle VNF instances at the time slot when their accumulative maintenance overhead is greater than θ. Clearly, at least n'_{ij} VNF instances of f_i should be kept in order to meet the end-to-end

delay requirements of running requests in $R_{ij}(t)$, where $n'_{ij}(t)$ can be calculated according to the delay requirements of requests in $R_{ij}(t)$ and Eq. (3).

To determine which idle VNF instances should be released to the system, we make use of historic offloading request traces (patterns) at each cloudlet to predict the number of VNF instances of each network function needed in the cloudlet in future. Specifically, we adopt an auto-regression method to predict the number of VNF instances $\hat{n}_{ij}(t)$ of f_i in cloudlet cl_j at the next time slot, using the information of the previous p time slots, assuming that the value of p is given.

$$\hat{n}_{ij}(t) = a_1 \cdot n_{ij}(t-1) + a_2 \cdot n_{ij}(t-2) + \cdots + a_p \cdot n_{ij}(t-p), \tag{11}$$

where $a_{p'}$ is a constant with $0 \le a_{p'} \le 1$, $\sum_{l=1}^{p} a_l = 1$, and $a_{p_1} \ge a_{p_2}$ if $p_1 < p_2$. Thus, the number $n_{ij}(t)$ of instances of f_i in cl_j that should be kept after time slot $t-1$ is

$$n_{ij}(t) = \max\{\hat{n}_{ij}(t), n'_{ij}(t)\}. \tag{12}$$

Similarly, if the number of VNF instances of a network function f_i is growing steadily at each time slot, more computing resources are needed to create its VNF instances, this will incur an extra cost since the instances are created one by one. To reduce the creation cost, we may proactively create the expected number of its VNF instances at once to meet its future need. This can be achieved, using the similar auto-regression method. That is, let $m_{ij}(t)$ be the number of new VNF instances of f_i in cloudlet cl_j added at time slot t. If the number of instances added since the last time slot t_0 exceeds a given threshold ξ, i.e., $\sum_{l=t_0}^{t} m_{ij}(l) \ge \xi$, then the predicted number of new instances $\hat{m}_{ij}$ of f_i at time slot t is

$$\hat{m}_{ij}(t) = b_1 \cdot m_{ij}(t-1) + b_2 \cdot m_{ij}(t-2) + \cdots + b_p \cdot m_{ij}(t-p), \tag{13}$$

where $b_{p'}$ is a constant with $0 \le b_{p'} \le 1$, $\sum_{l=1}^{p} b_l = 1$, and $b_{p_1} \ge b_{p_2}$ if $p_1 < p_2$. The number of instances of f_i after time slot $t-1$ installed in cl_j should be $\hat{m}_{ij}(t) + n_{ij}(t)$.

In all discussions so far, we assumed that there are sufficient resources at each cloudlet to create the expected number of VNF instances for each network function. However, if there are insufficient available resources at each cloudlet to meet the resource demands of different types of VNF instances, then questions arise as to which VNF instances should be created and how many of them to create? To fairly allocate the limited computing resources to VNF instance creations of different types of network functions, we adopt a strategy by proportionally scaling down the demanded number of VNF instances of each network function at each cloudlet as follows.

Let RC_j and DJ_j be the residual computing resource and the total computing resource demanded by different types of VNF instance creations in cloudlet cl_j, respectively. If $DI_j \le RC_j$, then all demanded VNF instances of different network functions will be created; otherwise, let $\chi_j = \frac{RC_j}{DI_j}$ be the ratio of available resource to the demanded resource in cl_j at time slot t. For each expected number of VNF instances of f_i, e.g., the total computing resource for creating $m_{ij}(t)$ VNF instances for f_i is $m_{ij}(t)C(f_i)$, we actually create $m'_{ij}(t) = \lfloor m_{ij}(t) \cdot \chi_j \rfloor$ VNF instances for f_i at cl_j.

The proposed online algorithm for the throughput maximization problem is given in `Algorithm 2`, which is referred to as algorithm `OL_MTS`.

Algorithm 2. OL_MTS(G)

Input: $G(C \cup V, E)$, $|C|$ cloudlets with resource capacity $cap_j(t)$, a set of network functions in $\mathcal{F}$, a set of requests $S(t)$ at each time slot t with $1 \le t \le T$, each idle VNF instance of f_i has a maintenance cost ϕ_{ij}, and the given release and creation cost thresholds θ and ξ.

Output: The number of requests admitted in the finite time horizon T.

$cost \leftarrow 0$; $M \leftarrow \emptyset$; /* the total cost of admitted requests to the system during a period T, and request assignment M */;
for all t with $1 \le t \le T$ **do**
  /* Stage one (a): Release some occupied resources by idle VNF instances if needed */
  $l_{ij} \leftarrow t_0$; /* The resource release procedure was performed at the last time slot t_0 with $t_0 < t$ */
  for each cloudlet cl_j **do**
    for each network function $f_i \in \mathcal{F}$ **do**
      if $\sum_{l=t-l_{ij}}^{t} \psi_{ij}(l) \cdot \phi_{ij} \ge \theta$ **then**
        Predict the number $\hat{n}_{ij}(t)$ of instances of f_i to be kept in cl_j by Eq. (11);
        Keep $\max\{n'_{ij}(t), \hat{n}_{ij}(t)\}$ VNF instances of f_i in cl_j;
        Release the occupied resources of $n_{ij}(t) - \max\{n'_{ij}(t), \hat{n}_{ij}(t)\}$ VNF instances of f_i in cl_j;
        Update the available resources at cl_j;
        $l_{ij} \leftarrow t$; /* reset the start time slot of the next idle VNF instances of f_i release in cl_j */
  /* Stage one (b): increase the number of VNF instances of f_i */;
  $I_{ij} \leftarrow t_0$; /* the number of instances of f_i was increased in the last time slot t_0 with $t_0 < t$ */
  for each cloudlet cl_j **do**
    for each $f_i \in \mathcal{F}$ **do**
      if $\sum_{l=t-I_{ij}}^{t} m_{ij}(l) \cdot \phi_{ij} \ge \xi$ **then**
        Predict the number of instances of f_i to be increased $\hat{a}_{ij}$ by Eq. (13);
  for each cloudlet cl_j **do**
    Let RC_j be the residual computing resource of cloudlet cl_j, and DI_j be the total computing resource needed by creating new instances;
    if $RC_j < DI_j$ **then**
      $\chi_j \leftarrow \frac{RC_j}{DI_j}$;
      For each network function f_i, there will be $\lfloor \chi_j \cdot (n_{ij}(t) + \hat{m}_{ij}(t)) \rfloor$ VNF instances in cl_j at time slot t;
    else
      There will be $n_{ij}(t) + \hat{m}_{ij}(t)$ VNF instances of f_i in cl_j at time slot t;
    Update the available resources at cloudlet cl_j;
    $I_{ij} \leftarrow t$; /* reset the start time slot of the next VNF instance increase of f_i in cl_j*/
  /* Stage two: perform request admissions*/
  $M \leftarrow \emptyset$; $cost \leftarrow 0$;
  for each t with $1 \le t \le T$ **do**
    M_t and $cost(t)$ delivered by invoking algorithm `OL_STS(`$G(t)$`)`;
    $M \leftarrow M \cup M_t$; $cost \leftarrow cost + cost(t)$;
return M is an assignment of requests, while $cost$ is the total admission cost to the system during a period T.

4.3 Algorithm Complexity Analysis

We first show that the solutions delivered by the proposed algorithms, Algorithms 1 and 2, are feasible. We then analyze the time complexity of the proposed algorithms as follows.

Theorem 2. *Given a mobile edge-cloud network $G(V \cup C, E)$, the number $n_{ij}(t)$ of VNF instances of each network function $f_i \in \mathcal{F}$ in each cloudlet cl_j with $1 \leq i \leq |\mathcal{F}|$ and $1 \leq j \leq m$, and a set $S(t)$ of requests at time slot t, there is an algorithm, Algorithm 1, for admitting the requests in $S(t)$ such that as many as requests are admitted while the operational cost is minimized. The proposed algorithm delivers a feasible solution. The time complexity of the algorithm is $O(d_{max}(|C| \cdot |\mathcal{F}| + |S(t)|)^3)$, where d_{max} is the maximum degree of nodes in $G(t)$, or the maximum number of cloudlets that any request can be assigned. Usually, the number of cloudlets in a mobile edge-cloud network is expected to be proportional to the logarithmic of the network size.*

Proof. We first show that the solution delivered by Algorithm 1 is feasible, i.e., for each admitted request, (i) its demanded resources and its end-to-end delay requirement are met; and (ii) there is no resource capacity violation in each cloudlet, as follows.

Following Algorithm 1, in the beginning of time slot 0, there are no requests in the system, the claim is true. We assume that all admitted requests in the first $t-1$ time slots meet conditions (i) and (ii). We now consider request admissions at time slot t. According to Algorithm 1, the request admissions at time slot t has a number of iterations.

Within iteration l with $1 \leq l \leq L \leq |C|$, all matched requests in $M_{t,l}$ of graph $G_l(t)$ will be admitted. Following the construction of $G_l(t)$, each admitted request in $M_{t,l}$ will meet conditions (i) and (ii); otherwise, no such an edge exists in $G_l(t)$. While all existing (running) requests sharing the VNF instances with a matched request in $M_{t,l}$ will still meet their individual delay requirements when the matched request is admitted. Then, the resources in each cloudlet are updated after allocating the demanded resources for the matched requests in $M_{t,l}$. The solution $\cup_{l=1}^{L} M_l$ thus is a feasible solution to the problem, where L is the number of iterations of the while loop in the algorithm.

We then analyze the time complexity of Algorithm 1. It can be seen that the construction of $G(t)$ takes $O(|X(t)| \cdot |C| \cdot |Y(t)|) = O(|\mathcal{F}| \cdot |C| \cdot |S(t)|)$, while the dominant time of the algorithm is to find a minimum weight maximum matching in $G_l(t)$ per iteration which takes $O((|C| \cdot |\mathcal{F}| + |S(t)|)^3)$ time. The number of iterations within each time slot is $O(d_{max})$, where d_{max} is the maximum degree of nodes in $G(t)$, i.e., the number of possible cloudlets that any request can be admitted while meeting its end-to-end delay requirement, thus $d_{max} \leq |C|$. □

Theorem 3. *Given a mobile edge-cloud network $G(V \cup C, E)$ in which cloudlets are co-located with some of the AP nodes, the number $n_{ij}(t)$ of existing VNF instances of each network function $f_i \in \mathcal{F}$ in each cloudlet cl_j with $1 \leq i \leq |\mathcal{F}|$ and $1 \leq j \leq |C|$, a set of requests $S(t)$ at each time slot t, and a given monitoring period T with $1 \leq t \leq T$, there is an online algorithm, Algorithm 2, for the throughput maximization problem, which delivers a feasible solution. The time complexity of the algorithm is $O(T \cdot d_{max}(|C| \cdot |\mathcal{F}|)^3 + \sum_{t=1}^{T} |S(t)|^3))$, where d_{max} is the maximum degree of nodes in $G(t)$, or the maximum number of cloudlets to which a request can be assigned.*

Proof. Following Algorithm 2, there are T time slots with each invoking Algorithm 1, it thus takes $O(T \cdot d_{max}(|C| \cdot |\mathcal{F}|)^3 + \sum_{t=1}^{T} |S(t)|^3)$ time. And the solution delivered is feasible, as within each time slot, all newly admitted requests and existing running requests at that time slot meet conditions (i) and (ii), by Theorem 2. The rest is to analyze the time complexity on VNF instance releases and creations at each cloudlet. It can be seen that idle VNF instance releases take $O(|C| \cdot |\mathcal{F}|)$ time as there are $|C|$ cloudlets and each cloudlet can accommodate the VNF instances of $|\mathcal{F}|$ different network functions. The time complexity of Algorithm 2 then follows. □

5 Online Algorithm for the Throughput Maximization Problem without End-to-End Delay Requirements

We now consider the throughput maximization problem without end-to-end delay requirements, by assuming that requests arrive in the system one by one without the knowledge of their future arrivals. We propose an online algorithm with a competitive ratio.

5.1 Primal-Dual Formulation

The basic idea of the proposed algorithm is to adopt the primal-dual dynamic updating method [5], [6], where *shadow price variables* for cloudlets in G are maintained to abstract the statuses of resource usages in cloudlets. A threshold is defined to decide whether a request can be admitted or not, by comparing the lowest shadow price with the given threshold. The shadow prices are updated accordingly if the request is admitted, for the sake of future request admissions. To this end, we formulate this special problem by an Integer Linear Programming (ILP) as follows.

Let S be the sequence of requests that arrive in the system one by one. For each arrived request $r_k \in S$, there is a decision variable x_{kj} to indicate whether request r_k is assigned to cloudlet cl_j if it is admitted. Recall that each cloudlet may have VNF instances instantiated already, or have available computing resource to create new VNF instances. For the sake of simplicity, we assume that each cloudlet cl_j has a duplicated *virtual cloudlet*, denoted by cl'_j, which represents the amount of computing resource in cl_j that can be used to create new VNF instances. Denote by C' the set of virtual cloudlets $C' = \{cl'_j \mid \forall cl_j \in C\}$. The objective of the ILP thus is

$$\textbf{LP}: \qquad \max \sum_{k=1}^{|S|} \sum_{j=1}^{|C \cup C'|} x_{kj}. \tag{14}$$

subject to the following constraints.

$$\sum_{j=1}^{|C \cup C'|} x_{kj} \leq 1, \qquad 1 \leq k \leq |S| \tag{15}$$

$$\sum_{k=1}^{|S|} x_{kj} \cdot \lambda_k \leq n_{ij} \cdot \mu_{ij}, \ \forall cl_j \in C, \forall 1 \leq i \leq |\mathcal{F}| \tag{16}$$

$$\sum_{k=1}^{|S|} x_{kj} \cdot \lambda_k \cdot RC^{unit} \leq cap_j, \quad \forall cl_j \in C' \tag{17}$$

$$\sum_{k=1}^{|S|} \sum_{j=1}^{|C'|} x_{kj} c_i^{ins} + \sum_{k=1}^{|S|} \sum_{j=1}^{|C|} x_{kj} \lambda_k \cdot \tau_k \cdot (C(f_i, cl_j) + \sum_{e \in P_{v_k, cl_j}} c_e) \leq B \tag{18}$$

$$x_{kj} \in \{0, 1\} \qquad \forall cl_j \in C \cup C', r_k \in S, \tag{19}$$

where B is a pre-defined budget for the cost of implementing all admitted requests, which is a constant. Constraints (15) guarantee that each request is assigned to at most one VNF instance (either an existing or newly created one). Constraints (16) indicate that the accumulative packet rate of the requests assigned to an existing VNF instance does not exceed its maximum packet (processing) rate, whereas Constraints (17) indicate that the amount of computing resource allocated to the creation of new VNF instances for admitted requests at each cloudlet should not exceed the amount of available computing resource of the cloudlet. Constraints (18) indicate the total cost of implementing all requests in S is no greater than a given budget B. In the proposed online algorithm, budget B can be set to a large value initially, and then easily tuned an appropriate value through binary search. Also, B is used in finding the dual of **LP** for online algorithm design. It determines the number of update steps of the shadow price variable related to the cost and represents the accuracy in the update of the shadow price. Thus, budget B affects the competitive ratio of the proposed online algorithm. This will be elaborated later in this section. Constraints (19) impose the integral constraint on each variable x_{kj}.

Let β_k, γ_{ij}, δ_j and η be the dual variables of constraints (15), (16), (17), and (18), respectively. The dual **DP** of **LP** is given below, and its objective is to

$$\begin{aligned}\textbf{DP}: \quad \min\ & \eta \cdot B + \sum_{k=1}^{|S|} \beta_k + \sum_{j=1}^{|C|} \sum_{i=1}^{|\mathcal{F}|} \gamma_{ij} \cdot n_{ij} \cdot \mu_{ij} \\ & + \sum_{j=|C|+1}^{|2C|} \delta_j \cdot cap_j,\end{aligned} \tag{20}$$

subject to the following constraint.

$$\begin{aligned}&\sum_{j=1}^{|C\cup C'|} \beta_k + \sum_{j=1}^{|C|} \sum_{i=1}^{|\mathcal{F}|} \gamma_{ij} \cdot \lambda_k + \sum_{j=|C|+1}^{2|C|} \delta_j \cdot \lambda_k \cdot RC^{unit} \\ &\quad + \sum_{j=1}^{|C'|} \eta \cdot c_i^{ins} + \sum_{j=1}^{|C|} \eta \cdot \lambda_k \cdot \tau_k \cdot A \geq 1,\end{aligned} \tag{21}$$

where $A = C(f_i, cl_j) + \sum_{e \in P_{v_k, cl_j}} c_e$.

The dual constraint (21) can be rewritten as

$$\begin{aligned}\sum_{j=1}^{|C\cup C'|} \beta_k \geq\ & 1 - \sum_{j=1}^{|C|} \sum_{i=1}^{|\mathcal{F}|} \gamma_{ij} \cdot \lambda_k - \sum_{j=|C|+1}^{2|C|} \delta_j \cdot \lambda_k \cdot RC^{unit} \\ & - \sum_{j=1}^{|C'|} \eta \cdot c_i^{ins} - \sum_{j=1}^{|C|} \eta \cdot \lambda_k \cdot \tau_k \cdot A.\end{aligned} \tag{22}$$

Notice that RC^{unit} is the amount of resource that is needed to process a unit packet rate, and μ_{ij} is the packet processing rate of a VNF instance of f_i.

5.2 Online Algorithm

We now describe the proposed online algorithm. Since the knowledge of future request arrivals is not given in advance, we need a smart admission policy to regulate the admission of the request being considered to minimize its impact on future request admissions. To this end, we define the shadow price P_j of each cloudlet cl_j, the maximum resource usage I^* of any request, and a constant Δ to guide resource reservation for future requests, where P_j denotes the marginal increase of strengthening the capacity and budget constraints (16), (17), and (18), i.e.,

$$P_j = \gamma_{ij} + \delta_j \cdot RC^{unit} + \frac{\eta \cdot c_i^{ins}}{\lambda_k} + \eta \cdot \tau_k \cdot A. \tag{23}$$

The maximum resource consumption I^* over all cloudlets for any request with unit packet rate is

$$I^* = \max\{RC^{unit}, c_i^{ins} + \tau_k \cdot A\}. \tag{24}$$

A constant Δ with an accuracy parameter ϵ to adjust resource reservation for a request then is

$$\Delta = I^*/\epsilon, \tag{25}$$

where $0 < \epsilon < 1$.

We then define *the admission control policy* of the online algorithm. For each arrived request $r_k \in S$, we identify a cloudlet with the minimum shadow price P_{j^*}, i.e., $cl_{j^*} = \arg\min_{cl_j \in C\cup C'} P_j$ for the admission of request r_k. Request r_k will be accepted if $P_{j^*} \leq 1$, or rejected otherwise. If it is admitted at cloudlet cl_{j^*}, then all dual variables of the constraints of **DP** will be updated, using the following updating rules:

$$\beta_k \leftarrow (1 - P_{j^*}), \tag{26}$$

$$\gamma_{ij^*} \leftarrow \gamma_{ij^*}\left(1 + \frac{\lambda_k}{n_{ij^*}\mu_{ij}}\right) + \frac{\lambda_k}{\Delta n_{ij^*}\mu_{ij}}, \tag{27}$$

$$\delta_{j^*} \leftarrow \delta_{j^*}\left(1 + \frac{\lambda_k \cdot RC^{unit}}{cap_{j^*}}\right) + \frac{\lambda_k}{\Delta \cdot cap_{j^*}}, \tag{28}$$

and

$$\eta \leftarrow \eta\left(1 + \frac{c^{ins} + \lambda_k \cdot \tau_k \cdot A}{B}\right) + \frac{c^{ins} + \lambda_k \cdot \tau_k \cdot A}{\Delta \cdot B}. \tag{29}$$

The detailed online algorithm, referred to as algorithm `OL_OBO`, is described in `Algorithm` 3.

Algorithm 3. OL_OBO $(G(t), S(t))$

Input: A mobile edge-cloud network $G(V \cup \mathcal{C}, E)$ and a set $\mathcal{F}$ of network functions provided by the network, a set S of requests that arrive one by one.

Output: Admit or reject each arrived request immediately, and an assignment of admitted requests to VNF instances in the cloudlets of C.

1: **while** an arrived request r_k **do**
2: $cl_{j^*} \leftarrow \arg\min_{cl_j \in C} P_j$;
3: **if** $P_{j^*} \geq 1$ **then**
4: Reject request r_k;
5: **else**
6: Assign request r_k to a VNF instance of type-i instances at cloudlet cl_{j^*};
7: Update dual variables β_k, γ_{ij}, δ_j, and η respectively, following rules in (26), (27), (28), and (29).

5.3 Algorithm Analysis

The rest is to analyze the competitive ratio of `Algorithm 3`. We first analyze the dual feasibility in the following lemma.

Lemma 1. *The dual feasibility of the dual variables is preserved, following the updating rules of (26), (27), (28), and (29).*

Proof. We show that all dual variables β_k, γ_{ij}, δ_j, and η are nonnegative, and the dual constraint (22) is met. For the dual variables, the four updating rules always keep them nonnegative, as they are set at zeros initially, and their values increase after each update. To show that constraint (22) is met, we consider whether an arrived request r_k will be admitted or rejected. If it is rejected, the admission control policy is not met, i.e., $P_{j^*} > 1$, which means that the right hand side of (22) is non-positive. Since β_k is set to 0 initially. Constraint (22) is met. Otherwise (request r_k is admitted), updating the dual variables makes the right hand side of constraint (22) become smaller, preserving the feasibility of the constraint. □

We then show an upper bound on the dual objective after the admission of r_k.

Lemma 2. *If request r_k is admitted, the objective value of the dual program increases by no more than $\lambda_k \cdot (1+\epsilon)$.*

Proof. Whenever request r_k is admitted, the dual variables corresponding to cloudlet cl_j in which it is admitted will be updated by rules (26), (27), (28), and (29), respectively. The objective value of the dual programming thus increases by

$$\begin{aligned}\Gamma_{r_k} &= \Big(\frac{\lambda_k \cdot \gamma_{ij}}{n_{ij}\cdot\mu_{ij}} + \frac{\lambda_k}{\Delta \cdot n_{ij}\cdot\mu_{ij}}\Big)\cdot n_{ij}\cdot\mu_{ij} \\ &\quad + \Big(\frac{\lambda_k \cdot RC^{unit}\cdot\delta_j}{cap_j} + \frac{\lambda_k}{\Delta\cdot cap_j}\Big)\cdot cap_j \\ &\quad + \Big(\frac{\eta\cdot c_i^{ins} + \eta\cdot\lambda_k\cdot\tau_k\cdot A}{B} + \frac{c_i^{ins}+\lambda_k\cdot\tau_k\cdot A}{\Delta\cdot B}\Big)B \\ &\quad + \lambda_k(1-P_j) \\ &= \lambda_k + \lambda_k\frac{2 + c_i^{ins}/\lambda_k + \tau_k\cdot A}{\Delta} \\ &= \lambda_k + \lambda_k\cdot\epsilon\cdot\frac{2+c_i^{ins}/\lambda_k+\tau_k\cdot A}{I^*} < \lambda_k(1+\epsilon).\end{aligned} \tag{30}$$

□

We finally analyze the constraint violations on the computing capacity of each cloudlet and the budget constraint in **LP** after considered the first k requests. Let $L_{ij}(k)$ be the total packet rate of the requests admitted by the n_{ij} VNF instances of type-i network function at cloudlet cl_j, and let $\gamma_{ij}(k)$ be the value of the dual variable γ_{ij} after the admission of request r_k. We then have the following lemma.

Lemma 3. *The violation of the computing capacity of each cloudlet cl_j in **LP** is at most by a multiplicative $O(1/\epsilon)$.*

Proof. Recall that for an arrived request r_k, $L_{ij}(k) = \gamma_{ij}(k) = 0$ initially for all cloudlets $cl_j \in C$ and all $f_i \in \mathcal{F}$ with $1 \le i \le |\mathcal{F}|$. If request r_k is admitted, the values of $L_{ij}(k)$ and $\gamma_{ij}(k)$ will be updated by

$$L_{ij}(k) = L_{ij}(k-1) + \lambda_k, \tag{31}$$

and

$$\gamma_{ij}(k) = \gamma_{ij}(k-1)\Big(1+\frac{\lambda_k}{n_{ij}\cdot\mu_{ij}}\Big) + \frac{\lambda_k}{\Delta n_{ij}\mu_{ij}}. \tag{32}$$

By induction analysis, we have

$$\begin{aligned}\gamma_{ij}(k) &\ge \frac{exp\frac{L_{ij}(k-1)}{n_{ij}\cdot\mu_{ij}} - 1}{\Delta}\Big(1+\frac{\lambda_k}{n_{ij}\cdot\mu_{ij}}\Big) + \frac{\lambda_k}{\Delta n_{ij}\mu_{ij}} \\ &= \frac{1}{\Delta}\Big(exp\Big(\frac{L_{ij}(k-1)}{n_{ij}\cdot\mu_{ij}}\Big)\Big(1+\frac{\lambda_k}{n_{ij}\cdot\mu_{ij}}\Big) - 1\Big) \\ &\approx \frac{1}{\Delta}\Big(exp\Big(\frac{L_{ij}(k-1)}{n_{ij}\cdot\mu_{ij}}\Big)exp\Big(\frac{\lambda_k}{n_{ij}\cdot\mu_{ij}}\Big) - 1\Big) \\ &= \frac{exp(\frac{L_{ij}(k)}{n_{ij}\cdot\mu_{ij}}) - 1}{\Delta}.\end{aligned} \tag{33}$$

Notice that $exp(x) \approx 1 + x$ is used to find an approximation of the inequality. Similar results can be found for the available computing resource cap_j and budget constraint B.

Let

$$1/Z_* = \min\{1, RC^{unit}, c_i^{ins}/\lambda_k, \tau_k\cdot A\}, \tag{34}$$

which means that

$$Z_* \ge 1. \tag{35}$$

Therefore, if $\gamma_{ij} > Z_*$, we have

$$\gamma_{ij} + \delta_j\cdot RC^{unit} + \frac{\eta\cdot c_i^{ins}}{\lambda_k} + \eta\cdot\tau_k\cdot A > 1, \tag{36}$$

which means that cloudlet cl_j will not admit request r_k.

Since the proposed online algorithm admits requests one by one, we have $\gamma_{ij} < Z_*$ in the last update of γ_{ij} for the processing of request r_{k-1}. Furthermore, the value of γ_{ij} increases by no more than I^* at each update, thus $\gamma_{ij} \le I^* + Z_*$. Incorporating Inequality (33), we have

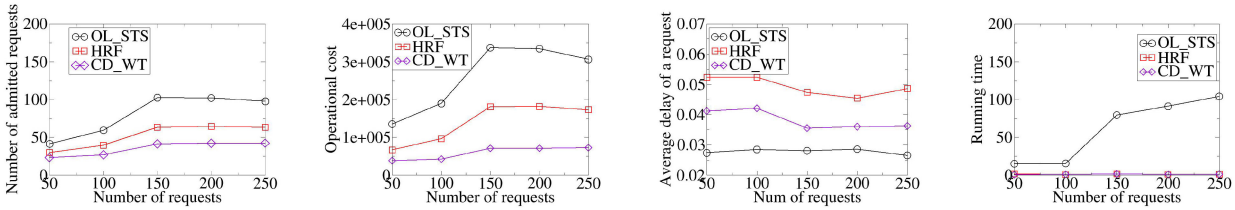
$$\frac{L_{ij}(k)}{n_{ij}\cdot\mu_{ij}} \le \log\left((I^*+Z_*)\Delta + 1\right) = O(1/\epsilon) \tag{37}$$

That is, the computing capacity violation of cloudlet cl_j by admitting request r_k is at most by a factor of $O(1/\epsilon)$. □

Theorem 4. *Given a mobile edge-cloud network $G(V \cup C, E)$ in which cloudlets are co-located with some of the AP nodes, the number n_{ij} of existing VNF instances of network function $f_i \in \mathcal{F}$ in each cloudlet cl_j with $1 \le i \le |\mathcal{F}|$ and $1 \le j \le |C|$, a set of requests S that arrive one by one without the knowledge of future request arrivals, there is an online algorithm with a provable competitive ratio of $\frac{1}{\lambda_{max}}(1-\epsilon)$, `Algorithm 3`, for a special case of the throughput maximization problem without the end-to-end delay requirement, where λ_{max} is the maximum packet rate among all requests and ϵ is a constant with $0 < \epsilon < 1$, i.e., $\lambda_{max} = \arg\max\{\lambda_k \mid 1 \le k \le |S|\}$. The running time of the algorithm is $O(|S|)$.*

Proof. By Lemma 2, let $|S| = k$, the objective value of the dual programming **DP** increases at most $\lambda_k(1+\epsilon)$ if the latest request r_k is admitted by a cloudlet. As a result, the objective value of the **LP** is at least $\frac{1}{\lambda_k(1+\epsilon)} \ge \frac{1}{\lambda_k}(1-\epsilon) \ge \frac{1}{\lambda_{max}}(1-\epsilon)$. The throughput delivered by the online algorithm is at least $\frac{1}{\lambda_{max}}(1-\epsilon)$ of the optimal one, where λ_{max} is the maximum packet rate of all requests.

The running time of `Algorithm 3` is analyzed as follows. In Step 2, the minimum value of P_j can be maintained in a min-heap, making this step $O(1)$ time. Also,



(a) The number of admitted requests (b) Operational cost (c) Average delay (d) Running time (seconds)

Fig. 2. Performance of algorithms OL_STS, HRF, and CD_WT by varying the number of requests from 50 to 250, while the number of cloudlets in the network is 10.

the updating of the dual variables takes O(1) time. As there are in total $|S|$ requests to be considered, the running time of the algorithm thus is $O(|S|)$. □

6 Performance Evaluation

In this section we evaluate the performance of the proposed algorithms by experimental simulations.

6.1 Experimental Settings

We assume that mobile edge-cloud networks are generated by the tool GT-ITM [7], with various network sizes from 50 to 250. Within each generated mobile edge-cloud network, there are 20 percent of its APs attached with cloudlets. Each cloudlet cl_j has a computing capacity cap_j randomly drawn from the range between 3 GHz and 10 GHz [8]. We assume that there are 10 different types of network functions, where the computing resource demand by each instance of a network function varies from 40 MHz to 400 MHz. We further assume that the link delay between two APs is a value between 2 milliseconds (ms) and 5ms [12]. The running time of an algorithm is based on a machine with a 3.7 GHz Intel i7-8700K CPU and 16 GiB RAM. Unless otherwise stated, these parameters are considered as the default settings.

The default number of requests per time slot is randomly drawn from the interval $[150, 250]$, and each request has a packet rate between 50 and 100 packets per second [17], and the tolerable delay requirement d_k of request r_k is a value between 0.01 and 0.3 seconds. The network function requested by each request is randomly selected from the 10 different types of network functions. Using the hourly price of a general purpose m3.xlarge Amazon EC2 instance as a reference, the operating cost is set to \$0.25 per MHz in each time slot, while the cost of instantiating a new function instance varies between \$20 to \$50. We assume that the cost of transferring a packet between two APs is proportional to their distance, thus the cost of transferring a packet along a network link varies between \$0.002 and \$0.005.

We evaluate the proposed algorithms against two benchmark algorithms. The first one is a greedy baseline which is described as follows. The greedy algorithm assigns each request r_k to the cloudlet with the highest rank, where the rank of a cloudlet is the product of its available numbers of VNF instances and the inverse of the operational cost of admitting r_k in the cloudlet. The rationale behind this method is to find a cloudlet with high numbers of available VNF instances and low admission cost, such that as many requests can be admitted while the total admission cost is minimized. We refer to this highest-rank-first baseline heuristic as HRF. The second benchmark algorithm is based on the basic idea of the algorithm in [14]. Notice that the problem in [14] is to minimize the weighted sum of the admission cost and response delay of admitted requests. Direct comparison between our algorithm with this algorithm may not be fair. Instead, we use the strategy of this algorithm to guide request admissions via a weighted function of the admission cost and the response delay per request, which is referred to as algorithm CD_WT. Also, algorithm OL_OBO will be compared with benchmark algorithm OL_BK that simply rejects requests when resources in cloudlets are not adequate to meet their resource demands.

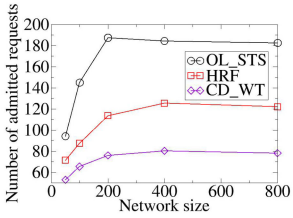
6.2 Algorithm Performance within a Single Time Slot

We first investigate the performance of the proposed algorithm OL_STS against that of algorithms HRF and CD_WT within a single time slot, by varying the number of requests from 50 to 250 while fixing the network size at 100, the number of cloudlets at 10, and creating some VNF instances at each cloudlet randomly as existing VNF instances.

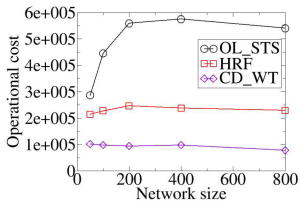
We can see from Figs. 2a and 2b that algorithm OL_STS admits more requests than algorithms HRF and CD_WT at a higher operational cost. This is because OL_STS assigns multiple requests to cloudlets simultaneously, multiple instances of network functions are placed among the cloudlets. Furthermore, since OL_STS admits more requests, it incurs a higher operational cost of implementing the admitted requests. Not surprisingly, algorithm CD_WT admits the least number of requests among the three comparison algorithms, because the metric it adopts only considers operational costs and delays. However, algorithm CD_WT does achieve the lowest operational cost and average delay per request, as shown in Figs. 2b and 2c.

Fig. 2c plots the average delay experienced by each admitted request. It can be seen that the average delays delivered by the three algorithms increase at first when the number of requests varies from 50 to 150, while the average delays decrease when the maximum number of requests reaches 250. The rationale behind this is that when the number of requests increases from 50 to 150, the number of admitted requests keeps increasing, thereby increasing the average delay in the waiting queues of cloudlets. However, when the number of requests increases from 150 to 250, as the waiting queues at cloudlets are already saturated. Fig. 2d illustrates the running times of the three algorithms, from which it can be seen that the running times increase with the number of requests.

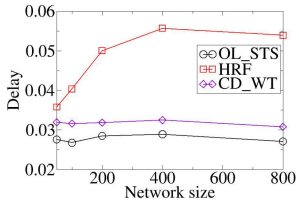
We then evaluate the impact of network size on the performance of the proposed algorithms by varying the network size from 50 to 800 while fixing the ratio of the



(a) The number of admitted requests

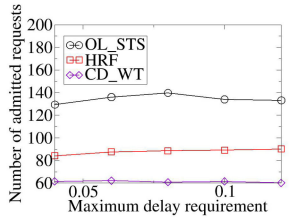


(b) The operational cost

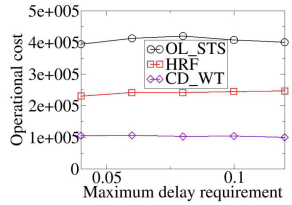


(c) Average delay

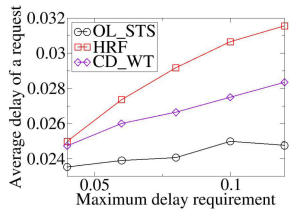
Fig. 3. Performance of algorithms OL_STS, HRF, and CD_WT by varying the network size from 50 to 800.



(a) The number of admitted requests



(b) Operational cost



(c) Average delay

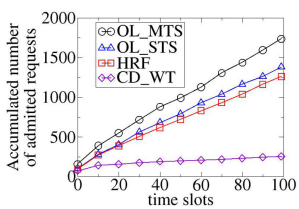
Fig. 4. Performance of algorithms OL_STS, HRF, and CD_WT when the maximum delay requirement of requests varies from 0.04 to 0.12.

number of cloudlets to the network size at 0.1, and creating some VNF instances for each type of network function at each cloudlet randomly. It can be seen from Fig. 3a that the number of requests admitted by algorithm OL_STS increases first from 50 to 200 and then decreases when the network size keeps increasing from 200 to 800. This is because a larger network means more cloudlets, and thus can admit more requests with the growth of its size. However, if the network size keeps increasing, some requests will reach their assigned cloudlets via longer routing paths, thereby increasing the probability of violating their delay requirements. This is evidenced by Fig. 3c, from which we can see the average delay of an admitted request increases with the growth of network size. Fig. 3b shows similar results on the operational cost.

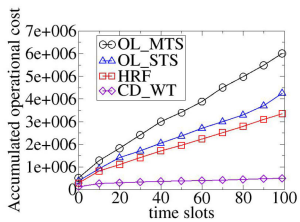
We finally study the impact of the maximum delay requirement among requests on the performance of the proposed algorithms by varying the maximum delay requirement of a request from 0.04 seconds to 0.12 seconds. It can be seen from Figs. 4a and 4b that more requests can be admitted if they have longer delay requirements, while the operational cost will increase due to the fact that more request admissions cause longer waiting delays in the queues of cloudlets. More importantly, the average delay per admitted request is increasing with the growth on the maximum delay requirement. In addition, algorithm OL_STS admits more requests with a higher operational cost than that by algorithms HRF and CD_WT in the long run, as shown in Figs. 4a, 4b, and 4c.

6.3 Performance Evaluation of the Online Heuristic

In the following we first evaluate the performance of algorithm OL_MTS within a finite time horizon that consists of 100 time slots. We assume that the number of requests at each time slot follows the Poisson distribution with a mean of 200, and each admitted request spans from 5 to 10 time slots randomly. Fig. 5a shows the number of requests admitted by algorithms OL_MTS, OL_STS without applying the prediction mechanism, HRF and CD_WT during the given time horizon, from which we can see that algorithm OL_MTS outperforms both algorithms HRF and CD_WT consistently.



(a) The accumulated number of admitted requests

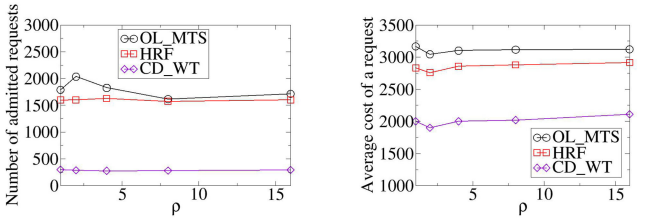


(b) The accumulated operational cost

Fig. 5. Performance of algorithms OL_MTS, OL_STS, HRF, and CD_WT in a time horizon of 100 time slots.

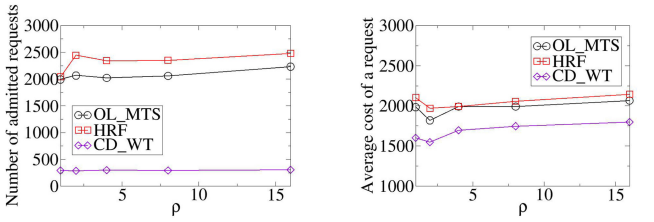
Fig. 5b plots the accumulative operational cost curves of the aforementioned algorithms. We can see that algorithm OL_MTS admits more requests than algorithms HRF and CD_WT at a higher operational cost compared to the other two algorithms. In particular, algorithm OL_MTS admits more requests than algorithm OL_STS without applying the proposed prediction mechanism. The reason is that algorithm OL_MTS pre-instantiates VNF instances if necessary and such pre-instantiation increases the admission probability of requests as it saves time and reduces delay through creating new instances, thereby meeting the delay requirements of admitted requests.

We then investigate the impact of important parameters a_i and b_j for controlling the prediction mechanism of algorithm OL_MTS, where a_i and b_j are the weights for the historical trace of existing VNF instances and newly created VNF instances in the last ρ time slots, respectively. We assume that the sequences for a_i and b_j are geometric sequences with a common ratio $1/2^{\rho}$ with $\rho \geq 1$, i.e., $a_{i+1} = \frac{a_i}{2^{\rho}}$ and $b_{j+1} = \frac{b_j}{2^{\rho}}$, and we vary the length of the historical trace ρ from 2 to 15. Fig. 6 shows the result. From Figs. 6a and 6b, we can see that there is no clear impact of ρ for the number of admitted requests; instead, the average operational cost of a request initially decreases when ρ increases from 1 to 2 first, but then increases with a longer historical trace from 2 to 16. The reason is that the length of the historical trace only affects the number of VNF instances



(a) The number of admitted requests

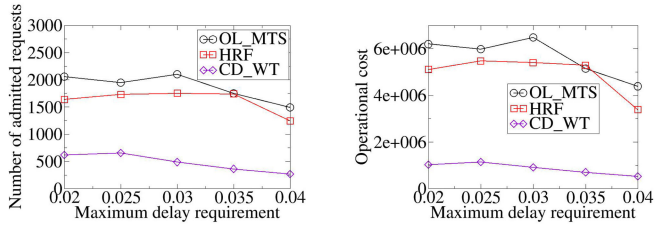
(b) The average operational cost per request



(c) The number of admitted requests

(d) The average operational cost per request

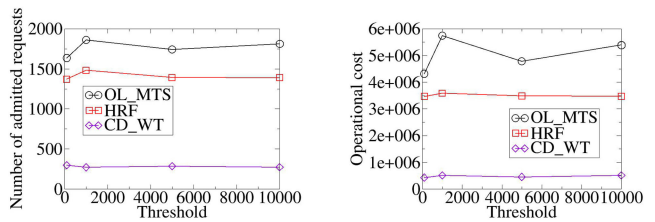
Fig. 6. Impact of a_i and b_j on the performance of algorithms OL_MTS, HRF, and CD_WT in a time horizon of 100 time slots, by varying p from 1 to 16.



(a) The number of admitted requests

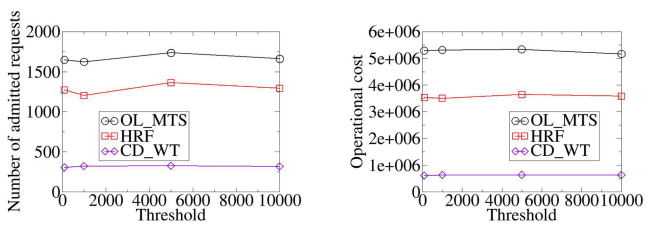
(b) The operational cost

Fig. 7. Performance of different algorithms, by varying d_k of each request r_k from 0.04 to 0.12 for a time horizon of 100 time slots.



(a) Numbers of admitted requests by varying threshold θ

(b) The operational cost by varying threshold θ



(c) Numbers of admitted requests by varying threshold ξ

(d) The operational cost by varying threshold ξ

Fig. 8. Performance of different algorithms for a time horizon of 100 time slots, by varying the idle cost threshold and the creation cost threshold from 100 to 10,000.

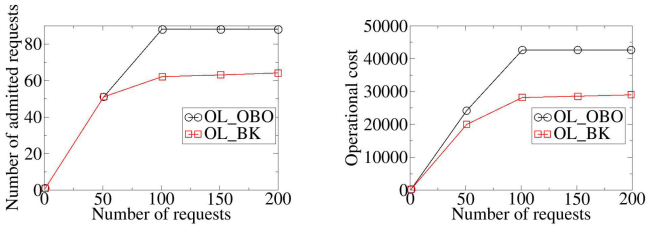
that should be created, and has no direct impact on the total amount of available resources that greatly affects the number of admitted requests. Furthermore, a larger value of p means larger values for the terms with lower indexes in each of the geometric sequences, implying that the recent historical data will have higher impact on the predicted number of VNF instances. This makes the prediction model accurately predict resource dynamics of cloudlets, thereby avoiding unnecessary creations of new VNF instances and reducing the VNF instantiation cost. However, with the increase of historical trace length of ρ, more weight is placed on older and less relevant data points. It can be seem that the best average operational cost per request is achieved when $\rho = 2$. Similar results can be found for b_j in Figs. 6c and 6d, respectively.

We thirdly evaluate the impact of the end-to-end delay requirement d_k of each request r_k on the performance of algorithms OL_MTS and HRF in a network with 100 APs and 10 cloudlets, by varying d_k from 0.04 seconds to 0.12 seconds. It can be seen from Fig. 7a that the longer the maximum delay per request is, the more requests the network can admit, while the operational cost is also increasing as shown in Fig. 7b.

We finally address the impact of the idle cost threshold θ of VNF instances and the creation cost threshold ξ on the performance of algorithm OL_MTS for a time horizon consisting of 100 time slots.

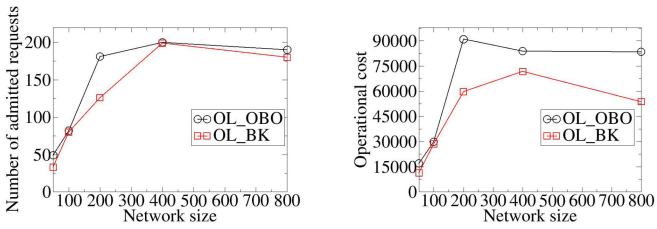
Figs. 8a, 8b, 8c illustrate the outcomes, by varying the idle cost threshold θ from 100 to 10,000 while fixing ξ at 1,000. Specifically, a small idle cost threshold θ will result in frequent invoking of the prediction mechanism. This means that idle VNF instances can be released back to the system more frequently, and therefore more requests can be admitted by utilizing the released cloudlet resources as shown in Fig. 8a when the idle cost threshold θ is set at 100. However, this usually incurs a higher operational cost due to the large number of requests admitted and the creation cost of new VNF instances for admitted requests by the released cloudlet resources. Furthermore, as shown in Fig. 8b, when the threshold increases from 5,000 to 10,000, the operational cost experiences a slight increase due to the increase on the maintenance cost of idle VNF instances.

Figs. 8d, 8e, 8f show the results by varying the creation cost threshold ξ from 100 to 10,000 while fixing θ at 1,000. We can see from Fig. 8d that the number of admitted requests increases from 100 to 5,000 steadily when ξ varies from 100 to 10,000, and then decreases when ξ reaches 10,000. The reason is that more new VNF instances can be created with a larger threshold ξ. However, when ξ is large enough (= 10,000 as shown in the figures), the prediction mechanism on the number of new VNF instances will be invoked rarely. This means that newly arrived requests may be frequently allocated to new VNF instances, and many such newly created VNF instances will became idle over time, thereby occupying resources and limiting the number of requests that can be admitted. In addition, as shown in Figs. 8e and 8f, the curves of the operational cost and average delay exhibit similar patterns to the curve of the number of admitted requests by the proposed algorithm. That is, more admitted requests implies a higher operational cost, as new VNF instances created for admitted requests may be placed into cloudlets far away from the locations of the admitted requests.



(a) The number of admitted requests (b) The operational cost

Fig. 9. Performance of online algorithms OL_OBO and OL_BK, by varying the number of requests from 50 to 200.



(a) The number of admitted requests (b) The operational cost

Fig. 10. Performance of online algorithms Online and Online_B, by varying the network size from 50 to 800.

6.4 Performance Evaluation of the Proposed Online Algorithm for the Problem without Delay Constraints

The rest is to evaluate the performance of the proposed online algorithm OL_OBO, by varying the number of requests from 50 to 200 while fixing the network size at 100 and the number of cloudlets at 10. It can be seen from Fig. 9a that algorithm OL_OBO outperforms the benchmark algorithm OL_BK, since a well-designed admission control policy is adopted. However, algorithm OL_BK performs simple admissions, only rejecting requests when insufficient resources are available in cloudlets, which could affect the admission of future requests. Also, we can see that the accumulated admitted requests increases first and then keeps steady afterwards. This is because the network is saturated when more requests are admitted. The operational costs delivered by different algorithms are shown in Fig. 9b. What follows is to investigate the impact of the network size on the performance of the proposed online algorithm, by varying it from 50 to 800 while fixing the ratio of the number of cloudlets to the network size at 0.1. From Fig. 10a, we can see that the number of requests admitted increases with the increase of the network size from 50 to 400, and stabilizes afterwards. The reason is that a large-size network has more cloudlets with more computing resource to admit more requests. However, the probability of routing user data traffic along longer routing paths increases too, which results in more user request rejections due to longer delays.

7 Conclusions

In this paper, we studied a novel task offloading problem in a mobile edge-cloud network, where each offloading task requests a specified network function service with a maximum tolerable delay, and different requests have different network function services from cloudlets in the network. We focused on maximizing the number of admissions of offloading tasks while minimizing the admission cost of admitted requests within a given time horizon. To this end, we devised an efficient online algorithm for the problem by opportunistically exploring existing VNF instances sharing among different requests or new VNF instance creations. We also developed an effective prediction mechanism to predict idle VNF instance releases and new VNF instance creations at different cloudlets for further cost savings. In addition, we also considered a special case of the problem without end-to-end delay requirements of requests, for which we devised an online algorithm with a provable competitive ratio. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results indicate that the proposed algorithms are promising.

Acknowledgments

The authors would like to thank the four anonymous referees and the associate editor for their expertise comments and constructive suggestions, which helped them improve the quality and presentation of the paper greatly. The work of Zichuan Xu is supported by the National Natural Science Foundation of China (Grant No. 61802048), the Fundamental Research Funds for the Central Universities in China (Grant No. DUT17RC(3)061), and the Xinghai Scholar Program in the Dalian University of Technology, China.

References

[1] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Select. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[2] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[3] Q. Fan and N. Ansari, "Workload allocation in hierarchical cloudlet networks," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 820–823, Apr. 2018.

[4] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2146–2153, Jun. 2018.

[5] M. X. Goemans and D. P. Williamson, "The primal-dual method for approximation algorithms and its application to network design problems," *Book Chapter of Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, Boston, 1997, pp. 144–191.

[6] L. Guo, J. Pang, and A. Walid, "Dynamic service function chaining in SDN-enabled networks with middleboxes," *IEEE 24th Int. Conf. Netw. Protocols*, 2016, pp. 1–10.

[7] (2018). [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

[8] Hewlett-packard company - enterprise computer server systems and network solutions, 2017. [Online]. Available: https://www.hpe.com/au/en/servers.html/, Accessed on: Apr. 25, 2017.

[9] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol 5, no 3, pp. 725–737, Oct.-Dec. 2017.

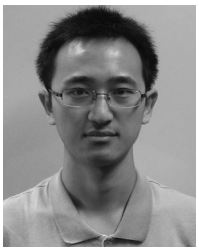
[10] M. Jia, W. Liang, and Z. Xu, "QoS-aware task offloading in distributed cloudlets with virtual network function services," *Proc. 20th ACM Int. Conf. Modelling Anal. Simul. Wireless Mobile Syst.*, 2017, pp. 109–116.

[11] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," *Proc. INFOCOM*, 2016, pp. 1–9.

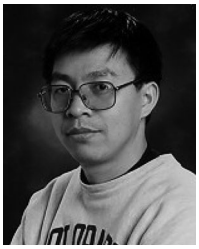
[12] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *J. Select. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[13] J. Kuo, S. Shen, H. Kang, D. Yang, M. Tsai, and W. Chen, "Service chain embedding with maximum flow in software defined network and application to the next-generation cellular network architecture," *Proc. INFOCOM*, 2017, pp. 1–9.

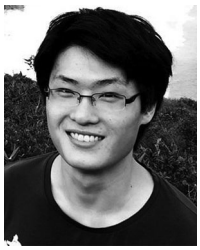
[14] Y. Li, Y. Chen, T. Lan, and G. Venkataramani, "MobiQoR: Pushing the envelope of mobile edge computing via quality-of-result optimization," *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 1261–1270.
[15] Y. Mao, C. You, J. Zhang, K, Huang, and K. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tuts.*, vol. 19, no. 4, pp. 2322–2358, Oct.-Dec. 2017.
[16] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, pp. 3746–3758, Sep. 2016.
[17] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.-Dec. 2009.
[18] Y. Song, S. S. Yau, R. Yu, X. Zhang, and G. Xue, "An approach to QoS-based task distribution in edge computing networks for IoT applications," in *Proc. IEEE Int. Conf. Edge Comput.*, 2017, pp. 32–39.
[19] Q. Xia, W. Liang, and W. Xu, "Throughput maximization for online request admissions in mobile cloudlets," in *Proc 38th Conf. Local Comput. Netw.*, 2013, pp. 589–596.
[20] Q. Xia, W. Liang, Z. Xu, and B. Zhou, "Online algorithms for location-aware task offloading in two-tiered mobile cloud environments," in *Proc 7th Int. Conf Utility Cloud Comput.*, 2014, pp. 109–116.
[21] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Capacitated cloudlet placements in wireless metropolitan area networks," *Proc. 40th Conf. Local Comput. Netw.*, 2015, pp. 570–578.
[22] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2016.
[23] B. Yang, W. K. Cai, Z. Xu, K. V. Katsaros, and G. Pavlou, "Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications," *IEEE Trans. Netw. Serv. Manage.*, vol. 15, no. 1, pp. 475–488, Mar. 2018.
[24] T. Zhang, "Data offloading in mobile edge computing: A coalition and pricing based approach," *IEEE Access*, vol. 6, pp. 2760–2767, 2018.



Zichuan Xu (M'17) received the BSc and ME degrees from the Dalian University of Technology, China, in 2008 and 2011, respectively, and the PhD degree from the Australian National University, in 2016, all in computer science. He was a research associate with the Department of Electronic and Electrical Engineering, University College London, United Kingdom. He currently is an associate professor with the School of Software, Dalian University of Technology, China. His research interests include cloud computing, software-defined networking, network function virtualization, wireless sensor networks, algorithmic game theory, and optimization problems. He is a member of the IEEE.



Weifa Liang (M'99-SM'01) received the BSc degree from Wuhan University, China, in 1984, the ME degree from the University of Science and Technology of China, in 1989, and the PhD degree from the Australian National University, in 1998, all in computer science. He is currently a professor with the Research School of Computer Science, Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, mobile edge computing (MEC), network function virtualization (NFV), software-defined networking (SDN), design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE and a member of the ACM.



Mike Jia received the BSc degree in mathematics and computer science from Imperial College London, United Kingdom, in 2013, and the honors degree with the first class honors in computer science from the Australian National University, in 2015. He is currently working toward the PhD degree in the Research School of Computer Science, Australian National University. His research interests include mobile cloud computing and software defined networks.

Meitian Huang received the BSc degree with the first class honors in computer science from the Australian National University, in 2015. He is currently working toward the PhD degree in the Research School of Computer Science, Australian National University. His research interests include software-defined networking, virtualized network function services, algorithm design and analysis, and cloud computing.

Guoqiang Mao received the PhD degree in telecommunications engineering from Edith Cowan University, in 2002. He currently is a professor of wireless networking, the director of the Center for Real-time Information Networks with the University of Technology, Sydney. He has published more than 100 papers in international conferences and journals, with more than 2,000 citations. His research interests include intelligent transport systems, applied graph theory and its applications in networking, wireless multihop networks, wireless localization techniques, and network performance analysis. He is a fellow of the IEEE, an editor of the *IEEE Transactions on Vehicular Technology* and a cochair of IEEE Intelligent Transport Systems Society Technical Committee on Communication Networks.