Mobility-Aware and Delay-Sensitive Service Provisioning in Mobile Edge-Cloud Networks

Yu Ma, Weifa Liang, *Senior Member, IEEE*, Jing Li, Xiaohua Jia, *Fellow, IEEE*, and Song Guo, *Fellow, IEEE*

Abstract—Mobile edge computing (MEC) has emerged as a promising technology to push the cloud frontier to the network edge, provisioning network services in proximity of mobile users. Serving users at edge clouds can reduce service latency, lower operational cost, and improve network resource availability. Along with the MEC technology, network function virtualization (NFV) is another promising technique that implements various network service functions as pieces of software in cloudlets (servers or clusters of servers). Providing virtualized network service for mobile users can improve user service experience, simplify network service deployment, and ease network resource management. However, mobile users move in networks arbitrarily, and different users usually request different services with different resource demands and delay requirements. It thus poses a great challenge to providing reliable and seamless virtualized network services for mobile users in an MEC network while meeting their individual delay requirements, subject to resource capacities on the network. In this paper, we focus on the provisioning of virtualized network function services for mobile users in MEC that takes into account user mobility and service delay requirements. We first formulate two novel optimization problems of user service request admissions with the aims to maximize the accumulative network utility and accumulative network throughput for a given time horizon, respectively. We then devise a constant approximation algorithm for the utility maximization problem. We also develop an online algorithm for the accumulative throughput maximization problem. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.

Index Terms—Mobile Edge computing, network function virtualization, VNF instance deployment, virtualized service provisioning, approximation and online algorithms, delay-sensitive request admission, utility gain maximization, user mobility, cloudlets or edge-clouds, resource allocations and provisioning in MEC, optimization problems

1 Introduction

Mobile devices, including smart phones and tablets, gain increasing popularity as communication tools of users for business, social networking, and personal entertainment. Meanwhile, more and more computation-intensive mobile applications with advanced features, including interactive online gaming, object recognition, and voice control, are developed for the convenience and experience of users. However, the processing of computing/storage intensive applications on portable mobile devices is heavily constrained by limited computation, storage and energy resources imposed on the mobile devices. Offloading computation-intensive applications to remote clouds with rich computing and storage resources can leverage the capabilities of mobile devices significantly. By doing so however may result in inevitable long response latencies, as clouds usually are remotely located from their end users, which degrades the user experience of using the services, especially for the services with stringent delay requirements [23]. Mobile Edge Computing (MEC) as a complement architecture of clouds, provides cloud services to mobile users at the network edge, thus shortens the response delay significantly. The MEC infrastructure can leverage the capability of mobile devices to offload their application services to nearby edge-clouds (cloudlets). Thus, the quality of service and energy consumption on mobile devices can be greatly improved [20].

In addition to MEC, another promising technique - Network Function Virtualization (NFV) has also been envisaged as the next-generation networking paradigm [4]. It leverages commodity servers to implement various network function services as software components in cloudlets, instead of purpose-specific hardware middleboxes. This introduces a new dimension of cost savings and deployment flexibility of network functions [26]. Each network function runs in a virtual machine, referred to as a VNF instance, hosted in a cloudlet. Taking a social VR application as an example [31], this application consists of user devices and virtualized network functions deployed for each corresponding user in cloudlets. A virtualized network function stores the user's personal data and the data processing logistics. It also deals with the state updates of the user and computation-intensive tasks, e.g., object recognition, visual scene calculations, and interactions with other users. User devices are only applied for rendering

- *Yu Ma, Weifa Liang, and Jing Li are with the Research School of Computer Science, The Australian National University, Canberra, ACT 2601, Australia. E-mail: {yu.ma, u6013763}@anu.edu.au, wliang@cs.anu.edu.au.*
- *Xiaohua Jia is with the Department of Computer Science, City University of Hong Kong83 Tat Chee Ave., Kowloon, Hong Kong. E-mail: csjia@cityu.edu.hk.*
- *Song Guo is with the Hong Kong Polytechnic University, Hong Kong. E-mail: song.guo@polyu.edu.hk.*

Manuscript received 18 Nov. 2019; revised 20 June 2020; accepted 29 June 2020. Date of publication 2 July 2020; date of current version 3 Dec. 2021. (Corresponding author: Weifa Liang.)
Digital Object Identifier no. 10.1109/TMC.2020.3006507

the video frames calculated by the VNF instances and tracking user behaviors.

Although implementing network functions as VNF instances in MEC can improve user service experience, simplify network service deployment, and ease network resource management, it poses great challenges: one is the computing resource capacity constraints on cloudlets, compared to traditional centralized clouds with virtually unlimited resources. It is of paramount importance to optimize the performance of MEC through judiciously allocating its resources to satisfy the service demands of as many users as possible. For the sake of convenience, in this paper we do not consider the storage resource as it usually is cheap and abundant, compared with the computing resource. Technically, the proposed algorithms can be easily modified to take into account additional types of resources in each cloudlet by considering a high-dimension bin packing problem. Another is that mobile users usually have stringent response delay requirements and high mobility in the network. How to provide reliable and seamless virtualized network services for them while meeting their stringent delay requirements and high mobility? and how to strive for a fine tradeoff between allocating resources to individual users and the overall service satisfactions among the users? In this paper, we will address the aforementioned challenges.

It must be mentioned that this paper is an extended version of a conference paper in [19]. The main differences between this journal version and its conference version are given as follows. We define a new VNF service provisioning problem to cope with user mobility in MEC networks, i.e., the online throughput maximization problem with the aim to maximize the accumulative number of user requests admitted for a finite time horizon, and an efficient solution to the problem is proposed too.

The novelty of the work in this paper lies in the thorough study of virtualized network service provisioning in MEC, by jointly considering user mobility and user service delay requirements. This study is the very first to explore the possibility to sacrifice some benefits of individual users by a tolerable extent for the admissions of more user service requests, by means of getting rid of some VNF service instances that are rarely used to make room for the deployment of VNF service instances for more user requests. A novel metric – network utility gain based on the submodular function, is proposed for this non-trivial tradeoff purpose, and an efficient approximation algorithm with a provable approximation ratio for the network utility maximization problem is developed, based on the proposed utility metric. Furthermore, a hybrid approach that combines both proactive and reactive VNF service provisioning is developed to deal with user mobility with the aim to maximize the number of user service requests admitted for the dynamic request admission scenario.

The main contributions of this paper are summarized as follows. We first formulate two optimization problems for the provisioning of delay-sensitive virtualized network services in MEC to highly movable users with the aim to either maximize the network utility gain for a given set of user requests, or maximize the accumulative network throughput for a given time horizon when user requests arrive one by one without the knowledge of future arrivals and the users move around in the network freely, subject to computing capacities on cloudlets in the MEC. We then show that both problems are NP-hard, and instead devise a constant approximation algorithm for the network utility maximization problem. We also develop an online algorithm for the online throughput maximization problem, by adopting a hybrid prediction mechanism for VNF instance placement. We finally evaluate the performance of the proposed algorithms through experimental simulations. Simulation results demonstrate that the proposed algorithms are promising and outperform their counterparts – the proposed benchmark algorithms.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 introduces the system model and Section 4 formulates novel mobility-aware and delay-sensitive user request admission problems mathematically. Section 5 devises an approximation algorithm for the network utility maximization problem, and Section 6 proposes an efficient online algorithm for the accumulative throughput maximization problem. Section 7 evaluates the performance of the proposed algorithms empirically, and Section 8 concludes the paper.

2 Related Work

As a key-enabling technology of 5G, MEC networks have gained tremendous attentions in the research community recently [20]. With the emergence of complicated and resource-hungry mobile applications, offloading user tasks to nearby cloudlets is an important means to reduce mobile devices' energy consumption and improve the service experience of mobile users.

Virtualized network service provisioning in clouds and MEC has been extensively addressed in literature [3], [7], [11], [12], [13], [14], [18], [36]. For example, Feng *et al.* [7] proposed an algorithm with performance guarantee for placing VNFs in distributed cloud networks and routing service flows among the placed VNFs under the constraints of the service function chains of requests. Their solution is achieved through a reduction to reduce the problem to a multi-commodity-chain flow problem on a cloud-augmented graph. He *et al.* [11] considered the problem of joint service placement and request scheduling in mobile edge clouds with the aim to maximize the network throughput, under resource capacity constraints. They proposed an approximation algorithm for a special case of the problem where both services and cloudlets are homogeneous. For the general case, they devised a greedy heuristic based on linear programming relaxation and randomized rounding. However, none of these mentioned works considered user mobility, which is fundamental in service provisioning to mobile users in MEC networks.

It must be mentioned that reducing the problem of VNF placement to cloudlets or its variants to the *facility location problem* is a common practice to solve service provision problems in MEC networks, which has extensive applications [27], [32], [33], [34], [35]. For example, Xu *et al.* [34], [35] dealt with the joint cloudlet placement and request scheduling problem by reducing the problem to the capacitated facility location problem. They devised scheduling algorithms to assign different cloudlets for the demanded services of different requests, by proposing efficient approximation and online algorithms with performance guarantees. Santoyo-Gonzalez *et al.* [27] studied the edge server placement problem in MEC network by developing a heuristic algorithm. They formulated a mixed

integer linear program solution, and then reduced the problem to a capacitated location-allocation problem. They aim to minimize service access latencies while taking into account capacity constraints and load balancing of servers. However, their solution by reducing the service provision problem to the Facility Location Problem is not applicable in this paper due the following two reasons. On one hand, in the facility location problem, each node (customer) can be served by any site (cloudlet) in the candidate site set, and each site may or may not have a capacity constraint. In this paper, each user has its exclusive set of VNF replicas at different locations, which means a VNF instance is associated with a single user and cannot serve other users. On the other hand, in the facility location problem, the demand of a user node can be distributed to multiple facilities with any proportion of its demands. In our problem, each user request is processed by a single VNF instance among a set of VNF instances assigned to the user. The number of service replicas deployed for a different user is different. For those locations with higher mobility probability, the replicas are very likely to be placed in nearby cloudlets, while for those with small probabilities may not be placed at all. To the best of our knowledge, the optimization problems in this work have not been studied previously, they are new problems to be investigated.

Several studies on user mobility in MEC networks have been taken recently [24], [28], [30]. For example, Ojima *et al.* [24] proposed a resource management framework for mobile edge computing networks, by applying user mobility prediction. They made use of the Kalman filter to predict the locations to which mobile users will move, and allocated resources based on the user mobility prediction. Guan *et al.* [10] studied the user mobility problem in a metropolitan MEC network. They devised a randomized algorithm to minimize the number of possible handovers between different MEC regions (thus minimizing possible delays), by dividing the metropolitan area into several disjoint clusters. Ouyang *et al.* [25] introduced the 'Follow Me Edge' concept to migrate the services among cloudlets to follow user mobility. They investigated a non-trivial tradeoff between user delays and costs incurred by service migration via the Lyapunov optimization technique. Lei *et al.* [16] proposed a link prediction model to predict user mobility dynamically. Their model combined deep neural networks in learning the distributed representations of networks as well as generative adversarial networks in generating high-quality weighted links. However, service migration usually is only applicable for delay-tolerant services, and is inappropriate for delay-sensitive applications such as real-time gaming, Virtual Reality (VR), and Augmented Reality (AR).

To provide reliable and seamless network services with stringent delay requirements while considering user mobility, one efficient approach is to replicate the VNF service requested by each mobile user to a certain number of strategic locations (cloudlets) to where the mobile user is very likely to move, in order to reduce the response delays to the requests from the user. The study of provisioning seamless services in MEC networks is very limited and in its infancy. The most relevant one is the one in [6], in which the authors proposed a proactive service migration approach for delay-sensitive applications to cope with user mobility. They formulated the problem as an Integer Linear Program (ILP) with the objective to minimize either the QoS degradation or the cost of replica deployment. This ILP solution however is not scalable when the problem size is large.

Unlike the aforementioned studies, in this paper we deal with mobility-aware and delay-sensitive service provisioning in MEC networks, through redundant VNF service instance placement in strategic locations. We are the very first to explore the possibility to sacrifice some benefits of individual users by a tolerable extent for the admissions of more user requests, by means of getting rid of some VNF service instances that are rarely used (or resulting in a low utility gain) to make room for the deployment of VNF service instances for more user requests. We introduce the network utility concept which is based on a submodular function for this non-trivial purpose, where the utility gain is that the overall satisfaction of users use the services provided for the amount of resource consumed. For the online case where users can move around within the network freely, we adopt a hybrid mechanism that combines both the proactive approach to pre-deploy VNF service instances to cloudlets to avoid the service response delays and the reactive approach to migrate existing VNF service instances.

3 System Model

In this section, we introduce the network model, and related notions and notations.

3.1 An MEC Network

We consider a metropolitan mobile edge-cloud computing network (MEC), which is modeled by an undirected graph $G = (\mathcal{AP} \cup V, E)$, where $\mathcal{AP}$ is a set of *access points* (APs) located at different locations in a monitoring metropolitan region, e.g., libraries, restaurants, gyms, hospitals, or shopping centers. A set V of cloudlets co-located with some of the APs, and the cloudlets have limited computing and storage resources to implement virtualized network functions (VNFs). Notice that the number $|V|$ of cloudlets usually is no greater than the number $|\mathcal{AP}|$ of APs. The communication delay between an AP and its co-located cloudlet is negligible. E is the set of *links* between APs. Each link $e \in E$ is a high-speed optical cable connecting a pair of APs. Denote by C_v the computing capacity of cloudlet $v \in V$. Each AP node is expected to cover a certain area in which mobile users can access the network through it. In case a mobile user is located in an overlapping region covered by multiple APs, the mobile user can choose the nearest AP for its connection. Other strategies can also be adopted such as choosing an AP with the strongest signal strength to connect. For simplicity, an AP node and its coverage area will be used interchangeably if no confusion arises. Fig. 1 is an illustrative example of an MEC network.

3.2 User Requests, User Mobility, and VNF Service Provisioning

We consider provisioning virtualized network function services to a set U of mobile users in an MEC $G(\mathcal{AP} \cup V, E)$, assuming that mobile users are highly movable and request delay-sensitive services. We term a location as a *sojourn location* of a user when the user moves to that location. Each user $j \in U$ at sojourn location l_j connects to the MEC

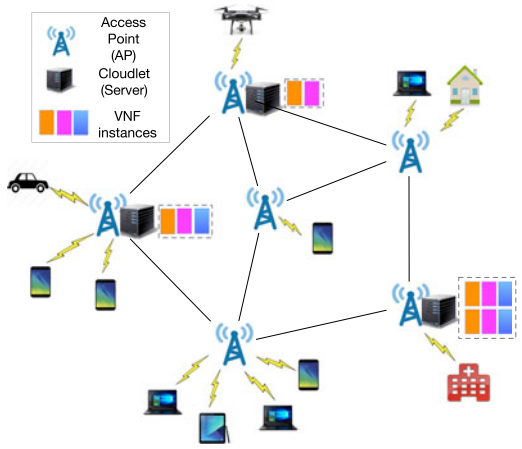


Fig. 1. An illustrative example of an MEC network that consists of 6 APs and 3 cloudlets co-located with some of the APs. The VNF instances of different services are deployed in the cloudlets.

network via its nearest AP l_j, through which the user requests for a VNF service that is represented by a tuple $r_j = \langle f_j, D_j, M_j \rangle$, where f_j is the type of VNF requested, D_j is the end-to-end delay requirement, and M_j is the mobility profile of mobile user j.

As users move frequently in the network, let p_{j,l_j} denote the probability of user j moving to location l_j. The mobility profile of user j thus is $M_j = \{p_{j,l_j} \mid l_j \in \mathcal{AP}\}$, which can be estimated based on the historical movement traces of the user, by leveraging data mining and machine learning techniques [6], [16], [22]. Let $\mathcal{AP}_j$ be the set of sojourn locations of user j in the network, i.e., $\mathcal{AP}_j = \{l_j \mid p_{j,l_j} > 0$ and $l_j \in \mathcal{AP}\}$. In a large-scale network, each user usually moves to only a very limited number of locations of the network. The cardinality $|\mathcal{AP}_j|$ thus is not large, and very likely to be a constant. We further assume that the mobility of different users is independent, i.e., the movement of one user does not impact the movement of others.

We assume that resources in cloudlets are virtualized, using container-based lightweight virtualization technologies, and thus can be allocated flexibly. Each VNF instance is a lightweight virtual machine. The VNF instance requested by a user $j \in U$ may not be necessarily in the cloudlet attached to AP l_j at which user j is located. For the sake of discussion simplicity, we assume that there is only one request associated with each mobile user. This simplification can be easily extended to deal with the case where there are multiple requests from a single user, by treating each of the requests as a request from a different *virtual user*, and all these virtual users in fact are the user. Since mobile users can move arbitrarily in the network, when a user moves to a new location, its end-to-end delay requirement D_j can be violated if its request is still served by the cloudlet at the previous location of the user.

3.3 Approaches for Dealing With User Mobility

Considering user mobility in MEC, it is vital to provide reliable and seamless VNF services to users with delay-sensitive applications. There are two approaches to achieve this goal: *the reactive approach* and *the proactive approach* [1], [20], [23].

In the reactive approach [17], [25], there is a corresponding (or a set of) VNF instance(s) in a cloudlet (a set of cloudlets) for each specified network function service of a user request. When the user moves to another location, either the VNF service instance of the user will be migrated to a nearby cloudlet at the new location, or a new VNF service instance for the user will be instantiated at a nearby cloudlet of the new location for the user to meet his delay requirement. This method however takes the extra cost and delay to migrate the existing VNF instance or instantiate a new VNF instance, which may result in a longer service disruption. Also, The implementation of this process consists of several stages [2], [29], such as resource availability check, file package building, testing and deploying, configuration setup, and dependency installations. The duration of the whole process takes from several seconds up to a few hours, depending on types of services.

In the proactive approach [6], the VNF instance of each service request is replicated to a number of replicas, and the replicas then are placed to the cloudlets at strategic locations (i.e., to which the user is very likely to move) in advance to meet the delay requirement of the user. Adopting this approach can save the latent service migration time significantly. It can be seen that service migration is only applicable for delay-tolerant applications, and inappropriate for delay-sensitive applications such as real-time gaming and augmented reality. This makes the proactive approach more appealing, even though it consumes more resources. In this paper, we will propose a hybrid method that opportunistically combines both mentioned strategies to cope with user mobility while meeting delay requirements of mobile users.

3.4 VNF Service Replications

To provide reliable and seamless virtualized network function services to mobile users while meeting their delay requirements, we adopt a *VNF service replication* strategy that pro-actively deploys a set of VNF replicas to a subset of cloudlets to respond to the request of the user. Each *VNF replica* $f_{j,v}$ is a VNF service instance of type f_j in cloudlet v for user j. Among the VNF replicas placed, one serves as the *primary VNF instance* and the rest serve as the *secondary VNF instances* of the user. Implementing VNF replicas in cloudlets consumes computing resources of cloudlets. Denote by $c(f_j)$ the amount of computing resource consumed by implementing a VNF instance f_j in any cloudlet.

3.5 Delay Requirements

In terms of Quality of Service (QoS) of each user, each request r_j of a user $j \in U$ has an end-to-end delay requirement D_j that specifies the maximum response time of request r_j, which consists of the access delay between user j and its AP, the transmission delay along a path between the AP and the cloudlet that contains the VNF instance for the packet processing, and the processing delay in a cloudlet for the requested service. Denote by $d_{acc}(l_j)$ the access delay between a user j and its AP l_j and $d_{pro}(f_{j,v})$ the processing delay of request r_j at a cloudlet v. Let $P(l_j, v)$ be the routing path for request r_j between its AP node l_j and cloudlet v if its service request will be processed at a VNF instance located at v. The data transmission delay along path $P(l_j, v)$ is $d(P(l_j, v)) = \sum_{e \in P(l_j,v)} d(e)$, where $d(e)$ is the transmission delay on link $e \in E$. The end-to-end delay $d(r_j)$ of request r_j

includes the transmission delay along a routing path, the access delay $d_{acc}(l_j)$, and the processing delay at cloudlet v, i.e., $d(r_j) = 2\big(d_{acc}(l_j) + d(P(l_j, v))\big) + d_{pro}(f_{j,v})$. To meet the end-to-end delay requirement of r_j, the admission of r_j must have $d(r_j) \leq D_j$, that is

$$d(P(l_j, v)) \leq \frac{1}{2}(D_j - 2 * d_{acc}(l_j) - d_{pro}(f_{j,v})). \quad (1)$$

Notice that the access delay $d_{acc}(l_j)$ between the mobile user j and AP l_j usually is not taken into account as part of D_j by several studies [6], [30], [31], because the value of $d_{acc}(l_j)$ varies over time and is determined by the bandwidth capacity of the AP and the number of users accessing the AP at the same time. In this paper, we adopt an expected delays of the AP access by its users based on their historical access traces to the AP.

3.6 Submodular Function and Network Utility Gain

Let Ω be a finite set and h a function with $h : 2^{\Omega} \mapsto R_0^+$. Function h is a non-decreasing submodular function if and only if it satisfies the following three properties:

1) $h(\emptyset) = 0$;
2) *Monotonic Increasing Property*: for $\forall X, Y \subseteq \Omega$ with $X \subseteq Y$, $h(X) \leq h(Y)$;
3) *Diminishing Return Property*: for $\forall X, Y \subseteq \Omega$ with $X \subseteq Y$ and $\forall a \in \Omega \setminus X$, $h(X \cup \{a\}) - h(X) \geq h(Y \cup \{a\}) - h(Y)$. In other words, $h(X) + h(Y) \geq h(X \cup Y) + h(X \cap Y)$.

Intuitively, mobile users move to different locations with different mobility probabilities. If a VNF instance is placed at each possible location for the user, some of the deployed VNF replicas will be barely used, thereby leading to poor resource utilization. Meanwhile, a user satisfaction on the network services provided by a network service provider can be captured by a submodular function, which means the increasing rate of user satisfaction on the service provisioning becomes gradually slower, when the probability of the delay requirement of the user being met increases from 0 to 100 percent. Thus, for any user, if his requested service can be met in most cases, then the network service provider may not necessarily place extra VNF replicas for him to meet the rest of his requirement. The amounts of saved resources can be used to accommodate more service requests for other users. We here explore the possibility to sacrifice some benefits of individual users by a tolerable extent for the admissions of more user requests, by means of getting rid of some VNF service instances that are rarely used to make room for the deployment of VNF service instances for more user requests. In other words, from the perspective of the service provider, it aims to meet user service demands in most cases while allowing few user service delay violations.

A submodular function $h(\cdot)$, which is to indicate the user satisfaction to VNF services provided by the MEC, can be adopted to strive for a fine tradeoff between individual user satisfactions on his requested service and the number of user requests admitted by the MEC. We refer to this submodular function as the *network utility gain function*, and we aim to maximize the accumulative utility gain among users.

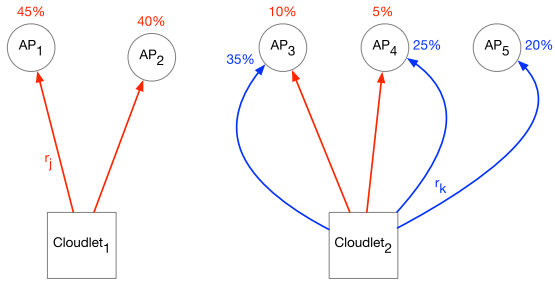


Fig. 2. An example of the network utility gain for requests r_j and r_k of users j and k, where both cloudlets have residual computing capacities 100. The mobility probabilities of user j towards different sojourn locations (marked with the red color) are 45 percent at AP_1, 40 percent at AP_2, 10 percent at AP_3, and 5 percent at AP_4, respectively, while the mobility probabilities of user k (marked as the blue color) towards different sojourn locations are 35 percent at AP_3, 25 percent at AP_4, and 20 percent at AP_5, respectively.

Fig. 2 is an example to illustrate the concept of the network utility gain.

We here adopt a submodular function $h(\cdot)$ $(= \log(x+1))$ as an illustrative example for the calculation of the network utility gain of the case in Fig. 2. Assuming that there are two mobile users j and k moving around in the network, and a VNF instance for request r_j or r_k has the computing resource demand of 100 computing units. In this case, only one VNF instance for either network functions can be deployed in each cloudlet. As illustrated by this figure, if one VNF instance for request r_j is deployed in $Cloudlet_1$, AP_1 and AP_2 will be covered for user j; while if one VNF instance for r_j is also deployed in $Cloudlet_2$, AP_3 and AP_4 will be covered. For user k, if a VNF instance for request r_k is deployed in $Cloudlet_2$, AP_3 AP_4, and AP_5 will be covered. Then, the marginal network utility gain by deploying a VNF instance in $Cloudlet_1$ for r_j is $\log(1.85) \approx 0.888$. If we further deploy a VNF instance for r_j in $Cloudlet_2$, the total network utility gain is $\log 2 = 1$ and request r_k cannot be admitted. Alternatively, if we deploy a VNF instance for request r_j at $Cloudlet_1$ and a VNF instance for request r_k at $Cloudlet_2$, then, the accumulative utility gain is $\log(1.85) + \log(1.8) \approx 1.74$, which achieves a higher network utility gain and both requests r_j and r_k can be admitted.

4 Problem Formulations

We consider two mobility-aware user request admission problems, by provisioning reliable and seamless VNF services to mobile users to meet their delay requirements. The first one is the network utility maximization problem, which aims to maximize the accumulative utility gain of all admitted requests through the deployment of VNF replicas at strategic locations for mobile users, subject to the resource capacities on cloudlets, given the mobility probability information of users and the set of users, assuming that the movement pattern of each user does not change in this monitoring period and we term this scenario as *the static snapshot scenario*. However, mobile users typically may move freely in the network and subject to changes of movement patterns in the long term. When a user moves to a new location, if none of its VNF replicas placed into cloudlets can provide its requested service without violating his delay requirement, then either a new VNF instance can be instantiated or

one of his existing VNF replicas can be migrated to a nearby cloudlet to meet his delay requirement, providing that there is sufficient computing resource in that cloudlet to accommodate the service resource demands. Meanwhile, existing users may leave the system when their requests finish and new users may arrive dynamically. The second problem in this paper is the online throughout maximization problem with the aim to maximize the number of user requests admitted for a finite time horizon, considering user mobility pattern changes over time. The precise definitions of these two problems are given below.

Definition 1. *Given an MEC $G = (\mathcal{AP} \cup V, E)$ with a set V of cloudlets, in which each $v \in V$ has computing capacity C_v, a set U of users moving around in the network with each user $j \in U$ having a service request $r_j = \langle f_j, D_j, M_j \rangle$, assuming that the movement profile of user $j \in U$ is given,* the network utility maximization problem *is to maximize the accumulative network utility gain of admitted requests, subject to computing resource capacities on cloudlets. In other words, the problem can be mathematically formulated as follows. For each user j, at most one VNF replica $f_{j,v}$ of its requested service can be placed in each cloudlet v, then all VNF replicas for user j constitute a set $\mathcal{F}_j = \cup_{v \in V}\{f_{j,v}\}$. Also, all VNF replicas in cloudlet v constitute a set $\mathcal{F}_v = \cup_{j \in U}\{f_{j,v}\}$. Let $\mathcal{F} = \cup_{j \in U, v \in V}\{f_{j,v}\}$ be the set of all VNF replicas of all user service requests in G.*

The routing path $P(l_j, v)$ for request r_j can be found, by finding a shortest path in G between its AP node l_j and cloudlet v co-located with another AP in terms of link delays. If there is a cloudlet v such that a VNF replica of r_j can be placed while meeting its delay requirement D_j when user j is located at AP l_j by Constraint (1), we say that AP l_j is *covered by cloudlet* $v \in V$. Let $S(j, v)$ be the set of APs covered by cloudlet v at which user j will sojourn. We construct a set system for each user $j \in U$ that consists of a collection of sets $S(j, v) \subseteq \mathcal{AP}_j$ of locations covered by deploying a VNF replica $f_{j,v} \in \mathcal{F}_j$ in cloudlet v. In other words, when user j moves to a location $l_j \in S(j, v)$, its delay requirement D_j can be satisfied by provisioning the requested service at cloudlet v.

Denote by $\mathbb{S}$ the set of VNF replicas deployed in the MEC network so far. Let $g'(f_{j,v} \mid \mathbb{S})$ be the marginal network utility gain by deploying a VNF replica $f_{j,v}$ into cloudlet v for the first time. Then,

$$g'(f_{j,v} \mid \mathbb{S}) = \sum_{l \in S(j,v) \setminus \cup_{v' \in \mathbb{S} \cap \mathcal{F}_j} S(j,v')} \big(h(x + p_{j,l_j}) - h(x)\big). \tag{2}$$

Recall that $h(x)$ is a submodular function, and x is the sum of mobility probabilities of user j to the locations covered by the VNF replicas of request r_j in $\mathbb{S}$ prior to the deployment of the VNF replica $f_{j,v}$. $S(j, v) \setminus \cup_{v' \in \mathbb{S} \cap \mathcal{F}_j} S(j, v')$ is the set of locations covered by $S(j, v)$ for the first time (by deploying the VNF replica $f_{j,v}$).

The network utility maximization problem is to deploy a subset $\mathcal{F}_{sol} \subseteq \mathcal{F}$ of VNF replicas one by one, and the accumulative network utility gain by Eq. (2) is maximized, subject to computing capacity on each cloudlet.

Definition 2. *Given a finite time horizon T, an MEC $G = (\mathcal{AP} \cup V, E)$ with a set $\mathcal{AP}$ of APs and a set V of cloudlets with each cloudlet v having computing capacity C_v. The time horizon T is divided into equal-length time slots. During each time slot $t \in T$, users can move around within the network or leave the network, and new users may arrive as well. Thus, users can be classified into three categories at each time slot t. One set of users leaving from the network; one set of users whose delay requirements can still be satisfied when they move around within the network; and one set $U(t)$ that consists of both newly arrived users and existing users whose delay requirements cannot be satisfied when they move around within the network. The set $U(t)$ of users request instantiating new VNF instances or migrating their existing VNF replicas to satisfy their delay requirements.* The online throughput maximization problem *is to maximize the accumulative number of users in set $U(t)$ admitted such that their delay requirements are met at each time slot t for the given time horizon T, subject to computing capacity on each cloudlet in G. Notice that there is no need to instantiate new VNF replicas or migrate existing VNF replicas for a user whose delay requirement can be met, and when a user leaves from the network, his VNF replicas will be removed and the occupied resource will be released back to the system.*

4.1 NP Hardness of Problems

In the following, we show that the two defined optimization problems are NP-hard.

Theorem 1. *The network utility maximization problem in $G = (\mathcal{AP} \cup V, E)$ is NP-hard.*

Proof. We prove the NP hardness of the network utility maximization problem by a reduction from a well known NP-hard problem - the knapsack problem that is defined as follows. Given a bin with capacity B, and a set $\mathcal{A}$ of items with each item $a_i \in \mathcal{A}$ having a specified size $size(a_i)$ and profit $profit(a_i)$, the problem is to find a subset of items packing into the bin such that the total profit is maximized, subject to the bin capacity B.

We show that an instance of the knapsack problem can be reduced to an instance of the network utility maximization problem. Specifically, the bin B corresponds to a cloudlet with capacity B, a set $\mathcal{A} = \{a_i\}$ of items corresponds to a set of VNF instances to be deployed. Each VNF instance a_i requires computing resource $size(a_i)$, and the amount of profit $p(a_i)$ received is the network utility gain $h(a_i)$ received when the VNF instance is deployed. We assume that there is only one cloudlet in the network. We aim to deploy a subset of VNF instances in the cloudlet so that the network utility gain is maximized, while the total size is upper bounded by the capacity B of the cloudlet. It is easy to see a solution to this special case of the network utility gain maximization problem is a solution to the knapsack problem. Thus, this theorem holds. □

Theorem 2. *The online throughput maximization problem in $G = (\mathcal{AP} \cup V, E)$ for a given time horizon T is NP-hard.*

Proof. We prove the NP hardness of the throughput maximization problem for one time slot by a reduction from a well known NP-hard problem - the Generalized Assignment Problem (GAP) that is defined as follows.

Given a set $\mathcal{A}$ of elements and a set $\mathcal{B}$ of bins where each bin $b_j \in \mathcal{B}$ has a capacity of $cap(b_j)$, and each element $a_i \in \mathcal{A}$ has a size $size(a_i, b_j)$ and a profit $profit(a_i, b_j)$ if

TABLE 1
Symbols

Symbols	Meaning
$G = (\mathcal{AP} \cup V, E)$	an MEC network with a set $\mathcal{AP}$ of access points, a set V of cloudlets, and a set E of links
v and C_v	a cloudlet $v \in V$ and computing capacity of v
e and $d(e)$	a link $e \in E$ and the transmission delay on link e
j, U and r_j	a user $j \in U$ with VNF service request r_j
l_j and $\mathcal{AP}_j$	a potential AP (location) l_j of user j and a set $\mathcal{AP}_j$ of potential locations of user j
f_j	the type of VNF requested by user j
$c(f_j)$	computing resource consumption by implementing a VNF instance of type f_j
$f_{j,v}$	a VNF instance of type f_j in cloudlet v for user j
D_j	the end-to-end delay requirement of user j
M_j	the mobility profile of user j
p_{j,l_j}	probability of user j moving to location l_j
$d_{acc}(l_j)$	the access delay between user j and its AP l_j
$d_{pro}(f_{j,v})$	the processing delay of request r_j at a cloudlet v
$P(l_j, v)$ and $d(P(l_j, v))$	the routing path of request r_j between AP l_j and cloudlet v and the transmission delay on the path
$d(r_j)$	the end-to-end delay of request r_j
$h(\cdot)$	a submodular function for the network utility gain
t and T	a time slot t in a finite time horizon T
$U(t)$	set of users requesting to instantiate new VNF instances or migrate existing VNF replicas to satisfy their delay requirements in time slot t
$\mathcal{F}_j$	all VNF replicas for user j
$\mathcal{F}_v$	all VNF replicas in cloudlet v
$\mathcal{F}$	all VNF replicas for all users in U
$\mathcal{F}_{sol}$	all VNF replicas deployed in MEC network
$S(j, v)$	the set of APs covered by cloudlet v at which user j will sojourn
$\mathbb{S}$	the set of VNF replicas deployed so far
$\mathbb{S}^{(k-1)}$ and $f^{(k)}$	the set of VNF replicas deployed so far, prior to the deployment of the kth VNF replica $f^{(k)}$
$c(\mathbb{S}_v^{(k)})$	accumulative computing resource consumed in cloudlet v by deploying the VNF replicas in $\mathbb{S}^{(k)}$
$g'(f_{j,v} \mid \mathbb{S})$	the marginal network utility gain by deploying a VNF replica $f_{j,v}$ into cloudlet v for the first time
$\psi(f^{(k)})$	a ratio defined for choosing the next VNF replica $f^{(k)}$ to be deployed
$g(\mathbb{S}^{(k)})$	the network utility gain by deploying a set of VNF replicas in $\mathbb{S}^{(k)}$
$\mathbb{S}'$ and $\mathbb{S}''$	two candidate solutions, each is a set of VNF replicas to be deployed
$g(\mathbb{S}')$ and $g(\mathbb{S}'')$	the network utility gain by deploying $\mathbb{S}'$ and $\mathbb{S}''$
ϕ and L	a time frame consists of L time slots
$f'_{j,v}$	an existing VNF instance for request r_j to be migrated
$G'(t)$ and $G'_i(t)$	the auxiliary graph constructed for VNF instantiations and migrations in time slot t, and $G'_i(t)$ is the auxiliary graph $G'(t)$ in the ith iteration
H_i	a maximum matching found in auxiliary graph $G'_i(t)$ in iteration i
$H(t)$	the union of all maximum matchings H_i in time slot t
$\hat{p}_{j,l}(\phi)$	the predicted mobility probability of user j towards location l_j in time frame ϕ
W and τ	W is the width of time frames for user mobility probability prediction, and τ is an iteration variable

element a_i is placed into bin b_j, *the Generalized Assignment Problem* (GAP) is to place as many elements as possible in $\mathcal{A}$ into the bins in $\mathcal{B}$ such that the profit sum of all placed elements is maximized, subject to the capacity of each bin in $\mathcal{B}$.

We show that an instance of the generalized assignment problem can be reduced to an instance of the throughput maximization problem for one time slot. Specifically, each bin $b_j \in \mathcal{B}$ corresponds to a cloudlet with capacity $cap(b_j)$, a set $\mathcal{A} = \{a_i\}$ of items corresponds to a set of requests to be admitted and processed by their VNF instances at a cloudlet. Each request a_i requires a VNF instance with specified computing resource $c(f_i)$ which is the size $size(a_i, b_j)$ of each item a_i in a bin b_j, and the profit $profit(a_i, b_j)$ received is 1 if it is admitted. Here we assume the delay requirement of each user request is not stringent, thus it can be admitted by provisioning a VNF replica in any cloudlet. We aim to admit a subset of requests by creating VNF instances for them assuming there are no existing VNF instances deployed for them, so that the number of user requests been admitted is maximized, while the size of each cloudlet is bounded by its capacity $cap(b_j)$. It can be seen that a solution to this special profit maximization problem is a solution to the generalized assignment problem, and the reduction is polynomial.

For each time slot $1 \leq t \leq T$, there is no knowledge of future request arrivals beyond time slot t, the online throughput maximization problem is to admit as many as requests for the period of T. Clearly, solving this online problem is at least as hard as solving its offline version - the throughput maximization problem for one time slot, the problem thus is NP-hard too. □

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

5 Approximation Algorithm for the Network Utility Maximization Problem

In this section, we study the network utility maximization problem. We first devise an approximation algorithm for

the problem, and then analyze the approximation ratio and time complexity of the proposed algorithm.

5.1 Algorithm

Notice that there is no need to determine to which cloudlet each VNF replica $f_{j,v}$ should be deployed, as it has already been scheduled to be placed at cloudlet v. The problem then is to find a subset of VNF replicas in each cloudlet v that can achieve the maximum network utility, subject to computing capacity on each cloudlet. However, VNF replicas in all cloudlets should be jointly considered as the sojourn locations covered by the VNF replicas at different cloudlets typically overlap with each other.

The proposed algorithm proceeds iteratively.

Let $\mathbb{S}^{(k-1)} = \{f^{(1)}, f^{(2)}, \ldots, f^{(k-1)}\}$ be the set of VNF replicas deployed in all cloudlets so far, prior to the deployment of the kth VNF replica $f^{(k)}$. $\mathbb{S}^{(0)} = \emptyset$ initially.

Denote by $c(\mathbb{S}_v^{(k-1)})$ the accumulative amount of computing resource consumptions in cloudlet v by the deployed VNF replicas in $\mathbb{S}^{(k-1)}$, that is,

$$c(\mathbb{S}_v^{(k-1)}) = \sum_{f^{(i)} \in \mathbb{S}^{(k-1)} \cap \mathcal{F}_v} c(f^{(i)}), \tag{3}$$

and $c(\mathbb{S}_v^{(0)}) = 0$ for each cloudlet $v \in V$.

The next VNF replica $f^{(k)}$ to be deployed in cloudlets is the VNF instance for a user j with the maximum ratio $\psi(f^{(k)})$ (assuming that $f^{(k)} = f_{j,v}$) among all the VNF replicas that have not been deployed yet, i.e., $f^{(k)} \in \mathcal{F} \setminus \mathbb{S}^{(k-1)}$, and the computing capacity of the targeting cloudlet v before deploying $f^{(k)}$ has not been violated, i.e., $c(\mathbb{S}_v^{(k-1)}) < C_v$, and $\psi(f^{(k)})$ is defined as

$$\psi(f^{(k)}) = \frac{g'(f^{(k)} \mid \mathbb{S}^{(k-1)})}{c(f^{(k)})}, \tag{4}$$

where $g'(f^{(k)} \mid \mathbb{S}^{(k-1)})$ is *the marginal utility gain* of VNF instance $f^{(k)}$ when a set $\mathbb{S}^{(k-1)}$ of VNF replicas have been deployed. Notice that the computing capacity of the targeting cloudlet v after the deployment of the VNF instance $f^{(k)}$ may be violated, i.e., $c(\mathbb{S}_v^{(k)}) = c(\mathbb{S}_v^{(k-1)}) + c(f^{(k)}) > C_v$. Consequently, cloudlet v may be overloaded, and this case will be dealt with later.

This procedure continues until no more VNF replicas can be deployed in any cloudlet without violating its computing capacity.

Let $g(\mathbb{S}^{(0)}) = 0$ and define the network utility gain $g(\mathbb{S}^{(k)})$ by $\mathbb{S}^{(k)}$ as follows.

$$g(\mathbb{S}^{(k)}) = g(\mathbb{S}^{(k-1)}) + g'(f^{(k)} \mid \mathbb{S}^{(k-1)}). \tag{5}$$

Since the set of deployed VNF replicas $\mathbb{S}^{(k)} \cap \mathcal{F}_v$ in some cloudlet v may violate its computing capacity C_v, the candidate solution $\mathbb{S}^{(k)}$ is partitioned into two disjoint subsets: $\mathbb{S}'$ and $\mathbb{S}''$, where set $\mathbb{S}''$ contains the VNF replicas that the addition of each of them to its assigned cloudlet v will violate the computing capacity constraint C_v of that cloudlet; and $\mathbb{S}'$ contains the rest of VNF replicas in $\mathbb{S}^{(k)}$, i.e., $\mathbb{S}' = \mathbb{S}^{(k)} \setminus \mathbb{S}''$. Clearly, $\mathbb{S}' \cap \mathbb{S}'' = \emptyset$ and $\mathbb{S}' \cup \mathbb{S}'' = \mathbb{S}^{(k)}$. The accumulative network utility gains $g(\mathbb{S}')$ and $g(\mathbb{S}'')$ obtained by sets $\mathbb{S}'$ and $\mathbb{S}''$ then are compared, and the larger one is chosen as the solution to the problem. The detailed algorithm is given in `Algorithm` 1.

Algorithm 1. An Approximation Algorithm for the Network Utility Maximization Problem

Input: An MEC $G = (\mathcal{AP} \cup V, E)$ with a set V of cloudlets each having computing capacity C_v, and a set U of users moving around in the network with each user $j \in U$ having an offloading task $r_j = \langle f_j, D_j, M_j \rangle$.

Output: A solution to maximize the network utility gain, by provisioning a set of VNF replicas in cloudlets for the users.

$\mathbb{S}^{(0)} \leftarrow \emptyset$; /* the set of deployed VNF replicas so far */
$g(\mathbb{S}^{(0)}) \leftarrow 0$; /* the accumulative network utility so far */
for each cloudlet $v \in V$ **do**
  $c(\mathbb{S}_v^{(0)}) \leftarrow 0$; /* the accumulative computing resource being occupied in cloudlet v so far */
end for
$k \leftarrow 1$;
$\mathcal{U} \leftarrow \cup_{j \in U, v \in V} \{f_{j,v} \mid S(j,v) \neq \emptyset\}$; /* $\mathcal{U}$ is the set of VNF replicas from all users that can cover some sojourn locations of users */
while $\mathcal{U} \neq \emptyset$ **do**
  Select a VNF replica $f^{(k)} \in \mathcal{U}$ that with the maximum ratio $\psi(f^{(k)})$ defined in Eq. (4);
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{f^{(k)}\}$;
  if $c(\mathbb{S}_v^{(k-1)}) < C_v$ for $f^{(k)} (= f_{j,v})$ in cloudlet v **then**
    $\mathbb{S}^{(k)} \leftarrow \mathbb{S}^{(k-1)} \cup \{f^{(k)}\}$;
    $g(\mathbb{S}^{(k)}) \leftarrow g(\mathbb{S}^{(k-1)}) + g'(f^{(k)} \mid \mathbb{S}^{(k-1)})$;
    $c(\mathbb{S}_v^{(k)}) \leftarrow c(\mathbb{S}_v^{(k-1)}) + c(f^{(k)})$;
    $k \leftarrow k + 1$;
  end if
end while
$\mathcal{F}_{sol} \leftarrow \emptyset$; /* the final solution */
Partition set $\mathbb{S}^{(k)}$ into two disjoint subsets: $\mathbb{S}'$ and $\mathbb{S}''$ with network utility gains $g(\mathbb{S}')$ and $g(\mathbb{S}'')$, respectively;
if $g(\mathbb{S}') \geq g(\mathbb{S}'')$ **then**
  $\mathcal{F}_{sol} \leftarrow \mathbb{S}'$;
else
  $\mathcal{F}_{sol} \leftarrow \mathbb{S}''$;
end if
return $\mathcal{F}_{sol}$ and $g(\mathcal{F}_{sol})$.

5.2 Algorithm Analysis

The rest is to analyze the approximation ratio and time complexity of `Algorithm` 1. We here abuse the notation OPT which will be used as both the optimal solution to the problem and the value of the optimal solution. We only pay an attention to cloudlets that do not have sufficient computing capacities to accommodate all VNF replicas of all requests, i.e., $C_v < \sum_{j \in U} c(f_j)$; otherwise ($C_v \geq \sum_{j \in U} c(f_j)$), VNF replicas of all requests can be deployed in cloudlet v, and the network utility gain achieved by cloudlet v is at least as the same as algorithm OPT. Without loss of generality, we assume that the computing capacity C_v of each cloudlet v is no greater than the total resource demand $\sum_{j \in U} c(f_j)$ of all users.

We have the following lemmas and theorem.

Lemma 1. *For any VNF replica $f^{(i)} \in \mathbb{S}^{(k)}$ delivered by `Algorithm` 1 and for any VNF replica $f^* \in OPT \setminus \mathbb{S}^{(k)}$, we have*

$$\psi(f^{(i)}) \geq \frac{g'(f^* \mid \mathbb{S}^{(k)})}{c(f^*)}. \tag{6}$$

Proof. If $OPT \setminus \mathbb{S}^{(k)} = \emptyset$, the lemma holds. Otherwise, when `Algorithm 1` chooses a VNF replica $f^{(i)}$ for placement, for any VNF replica $f^* \in OPT \setminus \mathbb{S}^{(k)}$ that has not been chosen by `Algorithm 1`, we have

$$\psi(f^{(i)}) = \frac{g'(f^{(i)} \mid \mathbb{S}^{(i-1)})}{c(f^{(i)})} \geq \frac{g'(f^* \mid \mathbb{S}^{(i-1)})}{c(f^*)} \tag{7}$$

$$\geq \frac{g'(f^* \mid \mathbb{S}^{(k)})}{c(f^*)}, \tag{8}$$

where Ineq. (7) holds due to that $f^* \in OPT \setminus \mathbb{S}^{(k)}$, $f^* \in OPT \setminus \mathbb{S}^{(i-1)}$, and Ineq. (8) holds due to the fact that the marginal utility gain $g'(\cdot)$ is diminishing with the deployments of more and more VNF replicas. □

Lemma 2. *The network utility gain $g(\mathbb{S}^{(k)})$ delivered by `Algorithm 1` through the deployment of a set $\mathbb{S}^{(k)}$ of VNF replicas is $g(\mathbb{S}^{(k)}) \geq \sum_{f^* \in OPT \setminus \mathbb{S}^{(k)}} g'(f^* \mid \mathbb{S}^{(k)})$.*

Proof. Let $f^*_{\max}$ be a VNF replica with the maximum ratio of $\frac{g'(f^* \mid \mathbb{S}^{(k)})}{c(f^*)}$ for a $f^* \in OPT \setminus \mathbb{S}^{(k)}$, i.e., $f^*_{\max} = \arg\max_{f^* \in OPT \setminus \mathbb{S}^{(k)}} \frac{g'(f^* \mid \mathbb{S}^{(k)})}{c(f^*)}$. We have

$$g(\mathbb{S}^{(k)}) = \sum_{i=1}^{k} g'(f^{(i)} \mid \mathbb{S}^{(i-1)}) = \sum_{i=1}^{k} \psi(f^{(i)}) \cdot c(f^{(i)}) \tag{9}$$

$$\begin{aligned}
&\geq \sum_{i=1}^{k} \frac{g'(f^*_{\max} \mid \mathbb{S}^{(k)})}{c(f^*_{\max})} \cdot c(f^{(i)}) \quad \text{by Ineq. (8)} \\
&= \frac{g'(f^*_{\max} \mid \mathbb{S}^{(k)})}{c(f^*_{\max})} \cdot \sum_{i=1}^{k} c(f^{(i)}) \\
&\geq \frac{g'(f^*_{\max} \mid \mathbb{S}^{(k)})}{c(f^*_{\max})} \cdot \sum_{f^* \in OPT \setminus \mathbb{S}^{(k)}} c(f^*) \qquad (10) \\
&\geq \sum_{f^* \in OPT \setminus \mathbb{S}^{(k)}} \frac{g'(f^* \mid \mathbb{S}^{(k)})}{c(f^*)} \cdot c(f^*) \\
&= \sum_{f^* \in OPT \setminus \mathbb{S}^{(k)}} g'(f^* \mid \mathbb{S}^{(k)}),
\end{aligned}$$

where Ineq. (10) holds due to the fact that for the optimal algorithm OPT, the accumulated usage of computing resource in any cloudlet v is no greater than its computing capacity C_v. However, the accumulated computing resource consumed in cloudlet v by deploying VNF replicas in $\mathbb{S}^{(k)} \cap \mathcal{F}_v$ is larger than C_v by `Algorithm 1`. □

Theorem 3. *Given an MEC $G = (\mathcal{AP} \cup V, E)$ with a set V of cloudlets, each cloudlet v has computing capacity C_v, and there is a set U of users with each $j \in U$ having a service request $r_j = \langle f_j, D_j, M_j \rangle$, there is a $\frac{1}{4}$-approximation algorithm, `Algorithm 1`, for the network utility maximization problem, which takes $O(|U|^2|V|^2|\mathcal{AP}|)$ time.*

Proof. Following Lemma 2, we have

$$\begin{aligned}
2 \cdot g(\mathbb{S}^{(k)}) &\geq g(\mathbb{S}^{(k)}) + \sum_{f^* \in OPT \setminus \mathbb{S}^{(k)}} g'(f^* \mid \mathbb{S}^{(k)}) \\
&\geq g(\mathbb{S}^{(k)} \cup OPT) \\
&\geq OPT,
\end{aligned} \tag{11}$$

where Ineq. (11) holds since $g(\mathbb{S}^{(k)})$ is the network utility achieved on $\mathbb{S}^{(k)}$, and $g'(f^* \mid \mathbb{S}^{(k)})$ is the marginal network utility gain of a VNF replica $f^* \in OPT \setminus \mathbb{S}^{(k)}$ when the set $\mathbb{S}^{(k)}$ of VNF replicas have been placed. Then, we have $g(\mathbb{S}^{(k)}) \geq \frac{1}{2} \cdot OPT$.

As $\mathbb{S}^{(k)} = \mathbb{S}' \cup \mathbb{S}''$ and the VNF replicas deployed in $\mathbb{S}'$ or $\mathbb{S}''$ do meet the computing capacity constraint on each cloudlet, we have

$$g(\mathbb{S}') + g(\mathbb{S}'') \geq g(\mathbb{S}' \cup \mathbb{S}'') = g(\mathbb{S}^{(k)}) \geq \frac{1}{2} \cdot OPT. \tag{12}$$

Then, either $g(\mathbb{S}') \geq \frac{1}{4} \cdot OPT$ or $g(\mathbb{S}'') \geq \frac{1}{4} \cdot OPT$, the solution delivered by `Algorithm 1` is at least $\frac{1}{4} \cdot OPT$ since the algorithm selects the larger one as its solution.

The rest is to analyze the time complexity of `Algorithm 1`. The algorithm consists of $O(|\mathcal{F}|) = O(|U| \cdot |V|)$ iterations. Within each iteration, calculating the marginal network utility gain for each VNF replica in $\mathcal{U}$ and identifying a VNF replica $f^{(k)} \in \mathcal{U}$ with the maximum ratio $\psi(f^{(k)})$ takes $O(|\mathcal{F}| \cdot |\mathcal{AP}|) = O(|U| \cdot |V| \cdot |\mathcal{AP}|)$ time. In total, it takes $O(|U|^2|V|^2|\mathcal{AP}|)$ time. Partitioning the solution $\mathbb{S}^{(k)}$ into two disjoint subsets $\mathbb{S}'$ and $\mathbb{S}''$ and calculating network utility gains for them take $O(|U| \cdot |V| \cdot |\mathcal{AP}|)$ time. `Algorithm 1` thus takes $O(|U|^2|V|^2|\mathcal{AP}|)$ time. □

6 Online Algorithm For the Online Throughput Maximization Problem

In the previous section, we considered the deployment of VNF replicas for user requests based on the mobility probabilities of each mobile user at different sojourn locations, and we strategically deployed the VNF service replicas of the user at his/her frequent sojourn locations to maximize the accumulative network utility gain of all user requests. However, once a user moves to a non-frequently sojourn location without his/her VNF service placement, the delay requirement of the user may be violated, as the requested service by the user will be performed at another location (cloudlet) where his VNF service instance has been placed.

In this section we consider a dynamic network service system, where existing users may leave and new users may arrive at any time. To allow the system to admit new user requests and to incorporate existing user mobility pattern changes, new VNF instances must be instantiated, or some existing VNF replicas must be migrated to meet the delay requirements of users. We focus on the provisioning of VNF services for mobile users in a dynamic service environment for a given time horizon, where user requests arrive one by one without the knowledge of future arrivals. We aim to maximize the accumulative network throughput for this monitoring period, by developing an online algorithm for the online throughput maximization problem.

6.1 Overview of the Online Algorithm

The basic idea behind the online algorithm is to divide the finite time horizon T under two different time scales: the pre-deployment of VNF replicas occurs in the beginning of each longer time scale to prevent the system overhead and instability. We term this time scale as *a time frame*. While each time frame can be further divided into equal numbers of *time slots*. The VNF instance instantiation and migration

are performed in the beginning of each time slot to meet user delay requirements and to support real-time user mobility. A hybrid approach adopting both proactive and reactive strategies is proposed. That is, within each time frame, we adopt the proactive strategy to pre-deploy VNF replicas, and within each time slot, we adopt the reactive strategy for VNF instance instantiation and migration. Without loss of generality, we assume that each time frame $\phi \in T$ consists of L time slots. In the following we propose a novel online algorithm for the online throughput maximization problem.

6.2 VNF Instance Instantiation and Migration at Each Time Slot t

Given a time slot $t \in \phi$, an MEC network $G = (\mathcal{AP} \cup V, E)$ with a set $\mathcal{AP}$ of APs and a set V of cloudlets, each cloudlet $v \in V$ has computing capacity C_v, and there is a set $U(t)$ of user requests (existing or newly arrived requests) in the beginning of time slot t. To admit the requests in $U(t)$, we either instantiate new VNF instances or migrate existing VNF replicas to the locations of these user requests in order to satisfy the delay requirements of their users. We aim to maximize the number of requests in $U(t)$ admitted by reducing the problem to a series of maximum matching problems in their corresponding auxiliary bipartite graphs that will be constructed later. Consequently, each matched edge in a maximum matching corresponds either an instantiation of a new VNF instance in the cloudlet that is another endpoint of the matched edge, or a migration of an existing VNF instance from its current cloudlet to another cloudlet that is another endpoint of the matched edge. The detailed reduction is as follows.

For each admitted request $r_j \in U(t)$ (in previous time slots), one of its VNF replica $f'_{j,v}$ will be migrated from its current location at cloudlet v if its removal will result in the minimum loss of the network utility gain, i.e., $f'_{j,v}$ is the VNF replica that will be least likely to be used among all VNF replicas $\mathcal{F}_j$ deployed for request r_j in the near future, that is

$$f'_{j,v} = \arg\min_{f_j \in \mathcal{F}_j} \big(g(\mathbb{S}) - g(\mathbb{S} \setminus \{f_j\})\big), \tag{13}$$

where $\mathbb{S}$ is the set of VNF replicas deployed in the network, and $\mathcal{F}_j$ is the set of VNF replicas of user request r_j.

For each newly arrived user request $r_j \in U(t)$, a new VNF instance for r_j will be instantiated in a nearby cloudlet with sufficient computing resource for the request to satisfy its delay requirement. To this end, a bipartite graph $G'(t) = (V, U(t), E')$ is constructed, where V is the set of cloudlets, $U(t)$ is the set of new and existing user requests in the beginning of time slot t. There is an edge in E' between a cloudlet node $v \in V$ and a request node $r_j \in U(t)$ if instantiating (or migrating) a VNF replica f_j of request r_j in cloudlet v does not violate its computing capacity and the delay requirement. Notice that when a user leaves from the network for good (or a given period threshold), all occupied resources by its VNF instances will be released back to the system in the end of that period.

The algorithm for maximizing the network throughput through VNF instance instantiation and migration at time slot t proceeds iteratively. Let $G'_1(t) = G'(t)$ initially. Within iteration i, a maximum matching H_i in graph $G'_i(t)$ is found. The demanded resource of a new VNF instance or migrating an existing VNF replica for each matched edge is allocated. All matched requests in H_i are then removed from $U(t)$. The available computing resource at each cloudlet is then updated, and the next auxiliary bipartite graph $G'_{i+1}(t)$ can be constructed similarly. This procedure continues until G'_{i+1} does not contain any edges. Denote by I the number of iterations of the above proposed algorithm. The union of all maximum matchings $H(t) = \bigcup_{i=1}^{I} H_i(t)$ then forms a solution to the network throughput maximization problem for requests in $U(t)$ at time slot t. Notice that, the proposed algorithm can be extended to a general case of the problem, where both VNF instantiation and migration costs can be taken into account, by finding a *minimum weight maximum matching*, instead of a *maximum matching*, in each weighted auxiliary graph $G'_i(t)$ where the VNF migration cost and the instantiation cost can be assigned to the corresponding edges in the auxiliary graph. Such a modification is straightforward, the only difference is the running time, finding a maximum matching in a bipartite graph with n nodes and m edges takes $O(\sqrt{n}m)$ time, while finding a maximum (or minimum) weighted maximum matching in the bipartite graph takes $O(n^3)$ time.

The detailed algorithm is given in `Algorithm` 2.

Algorithm 2. An Algorithm for Throughput Maximization of a Set $U(t)$ of Requests in Time Slot t

Input: An MEC $G = (\mathcal{AP} \cup V, E)$ with a set $\mathcal{AP}$ of APs and a set V of cloudlets each having computing capacity C_v, a set $U(t)$ of user requests that request instantiating new VNF instances or migrating existing VNF replicas in order to satisfy their delay requirements when they move around in G.

Output: A solution to maximize the number of admitted requests in set $U(t)$, by instantiating new VNF instances or migrating existing VNF replicas, subject to computing capacity on each cloudlet in G.

1: Identify the VNF instance $f'_{j,v}$ to be migrated for each existing request r_j in set $U(t)$ by Eq. (13);
2: Construct a bipartite graph $G'(t) = (V \cup U(t), E')$;
3: $i \leftarrow 1$;
4: $G'_1(t) \leftarrow G'(t)$;
5: $H(t) \leftarrow \emptyset$;
6: **while** $G'_i(t)$ contains edges **do**
7: Find a maximum matching $H_i(t)$ in $G'_i(t)$, by an algorithm for maximum matching [5];
8: Instantiate new VNF instances or migrate existing VNF replicas from their current host cloudlets to their target cloudlets through the matched edges in $H_i(t)$;
9: Update the residual amount of computing resource in each cloudlet v which can be derived from the corresponding matched edge in $H_i(t)$;
10: Remove all requests that have been matched in $H_i(t)$ from set $U(t)$;
11: $H(t) \leftarrow H(t) \cup H_i(t)$;
12: $i \leftarrow i + 1$;
13: Construct the next auxiliary bipartite graph $G'_{i+1}(t)$;
14: **end while**
15: Release the occupied resources by their VNF instances to the system when users leave from the network for good (or for a given period larger than a threshold in the end of time slot t.
16: **return** $H(t)$.

6.3 Online Algorithm

We now propose an online algorithm for the online throughput maximization problem for a given time horizon T. The algorithm runs in two time scales: time frames and time slots. Within each time frame $\phi \in T$, VNF replicas are pre-deployed based on the mobility probability of existing mobile users, while in each time slot $t \in \phi$, VNF instantiation and migration are performed to maximize the number of user requests admitted, where existing users may leave and new users may arrive, and users can move around in the network freely.

We start with the user mobility prediction. Specifically, we adopt the auto-regression method [8] to predict the mobility probability $\hat{p}_{j,l}(\phi)$ of each user j sojourning at a location $l_j \in \mathcal{AP}$ in the next time frame $\phi \in T$, using historical mobility traces of user j. Let W be width of a time window (the number of time frames per time window) that we use for prediction. The sojourn duration $\delta_{j,l}^{(\phi-\tau)}$ of user j sojourns at location l_j (in terms of the number of time slots) in time frame $\phi - \tau$ is tracked with $1 \leq \tau \leq W$. The predicted mobility probability $\hat{p}_{j,l}(\phi)$ of user j towards location l_j is

$$\hat{p}_{j,l}(\phi) = \sum_{\tau=1}^{W} \alpha_\tau \cdot \frac{\delta_{j,l}^{(\phi-\tau)}}{L}, \tag{14}$$

where α_τ is a constant with $0 \leq \alpha_\tau \leq 1$, $\sum_{\tau=1}^{W} \alpha_\tau = 1$, $\alpha_{\tau_1} \geq \alpha_{\tau_2}$ if $\tau_1 < \tau_2$, and L is the length of each time frame ϕ.

We then perform the pre-deployment of VNF replicas at each time frame. We finally conduct VNF instance instantiation and migration within each time slot for each given time frame. This procedure continues until the given time horizon is covered. The online algorithm is given in `Algorithm` 3.

Algorithm 3. An Online Algorithm for the Online Throughput Maximization Problem

Input: An MEC $G = (\mathcal{AP} \cup V, E)$ with a set $\mathcal{AP}$ of APs and a set V of cloudlets with each having computing capacity C_v, and a set U of users moving around in G for a given time horizon T.

Output: A solution to maximize the accumulative number of user requests admitted for the time horizon T, assuming that T is partitioned into the number of equal time frames.

1: **for** each time frame $\phi \in T$ **do**
2: Predict the user mobility probability $\hat{p}_{j,l}(\phi)$ of each user j moving to location $l_j \in \mathcal{AP}$ in the next time frame ϕ by Eq. (14);
3: Deploy a set $\mathbb{S}$ of VNF replicas for users in U to maximize the network utility gain, by invoking `Algorithm` 1;
4: **for** each time slot $t \in \phi$ **do**
5: /* A set of user requests $U(t)$ requesting to instantiate new VNF instances or migrate existing VNF replicas to satisfy their delay requirements when their users move around in the network */
6: Perform VNF instance instantiations and migrations, by invoking `Algorithm` 2.
7: **end for**
8: **end for**

6.4 Algorithm Analysis

We now analyze the performance of `Algorithms` 2 and 3 as follows.

Lemma 3. *Given an MEC $G = (\mathcal{AP} \cup V, E)$ with a set V of cloudlets with each $v \in V$ having computing capacity C_v, a set $U(t)$ of user requests that request instantiating new VNF instances or migrating existing VNF replicas to satisfy their delay requirements when mobile users move to new locations, there is an algorithm, `Algorithm` 2, to maximize the network throughput for a set $U(t)$ of user requests in time slot t. The algorithm takes $O(|U(t)| \cdot |V \cup U(t)|^{2.5})$ time.*

Proof. Following `Algorithm` 2, within each iteration, a VNF instance can be instantiated or migrated only if there is sufficient computing resource in the target cloudlet. The computing resource of a migrated VNF replica in its previous cloudlet will be released back to the system. There are I iterations in total for VNF instance instantiation and migration, and the computing capacity constraint on each cloudlet is not violated at any iteration, the solution delivered by `Algorithm` 2 thus is feasible.

The time complexity of `Algorithm` 2 is analyzed as follows. In each iteration, the construction of an auxiliary bipartite graph takes $O(|V| \cdot |U(t)|)$ time. A maximum matching finding in the auxiliary graph takes $O(|V \cup U(t)|^{2.5})$ time [5]. `Algorithm` 2 thus takes $O(|U(t)| \cdot |V \cup U(t)|^{2.5})$ time, as there are $I = O(U(t))$ iterations. □

Theorem 4. *Given a finite time horizon T, an MEC $G = (\mathcal{AP} \cup V, E)$ with a set $\mathcal{AP}$ of APs and a set V of cloudlets with each $v \in V$ having computing capacity C_v, a set U of mobile users with VNF service requests moving around in the network, a set $U(t)$ of users that request instantiating new VNF instances or migrating existing VNF replicas to satisfy their delay requirements at each time slot t, there is an online algorithm, `Algorithm` 3, for the online throughput maximization problem, which takes $O(|U|^2|V|^2|\mathcal{AP}|)$ time for the pre-deployment of VNF replicas for each time frame $\phi \in T$, and $O(|U(t)| \cdot |V \cup U(t)|^{2.5})$ time for VNF instance instantiation and migration at each time slot $t \in \phi$.*

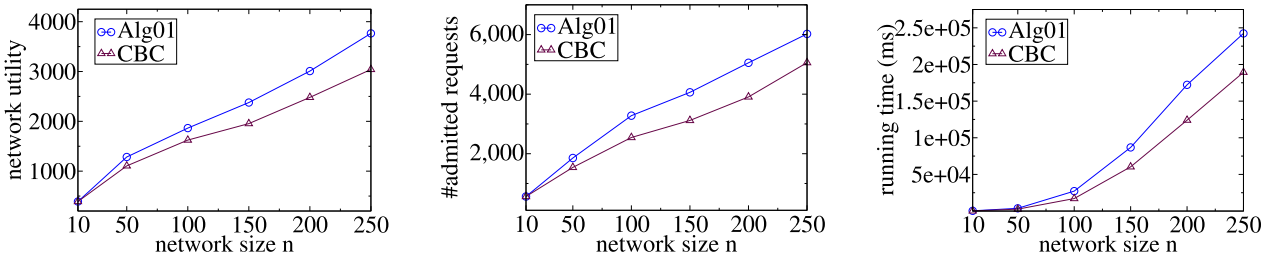
The proof of `Theorem` 4 is omitted, since the running time of `Algorithm` 3 follows from `Theorem` 3 and `Lemma` 3 directly.

7 Performance Evaluation

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

7.1 Experimental Environmental Settings

We consider an MEC $G = (\mathcal{AP} \cup V, E)$ consisting of from 10 to 250 nodes. We adopt a commonly used tool GT-ITM [9] to generate network topologies. 10 percent of AP nodes are co-located with cloudlets. The computing capacity of each cloudlet varies from 4,000 MHz to 8,000 MHz [13]. The transmission delay on a link varies from 2 ms to 5 ms [15]. The computing resource demand of the VNF service instance of each request is set from 10 MHz to 100 MHz [13]. For each user j, the number of possible mobility locations is no more than 30 percent of the number of APs in the network, and its end-to-end delay requirement D_j varies from 10 ms to 100 ms [21]. The submodular network utility function $h(\cdot)$



(a) The accumulative network utility gain (b) The number of requests admitted (c) The running times

Fig. 3. Performance of different algorithms by varying network size from 10 to 250.

adopted is $\log(x+1)$. Note that the proposed algorithms still work if other submodular network utility functions are adopted, and the performance trends of the algorithms with different submodular functions are similar with each other. The value in each figure is the mean of the results out of 50 MEC network instances of the same size. The running time of an algorithm is obtained based on a machine with 3.4 GHz Intel i7 Quad-core CPU and 16 GB RAM. Unless otherwise specified, these parameters will be adopted in the default setting.

We first evaluate the performance of the proposed approximation algorithm `Algorithm 1` against a benchmark heuristic `CBC` (cloudlet by cloudlet), which deploys a VNF replica with the maximum ratio of network utility gain to its computing resource consumption in a cloudlet, providing that that cloudlet has sufficient residual computing resource for the placement. After all VNF replicas in a cloudlet have been examined, it will move to the next cloudlet. We then evaluate the performance of `Algorithm 2` against a baseline algorithm `Greedy`, where algorithm `Greedy` selects a request with the minimum resource demand and instantiates a new VNF instance or migrate an existing VNF instance to a cloudlet that has the least routing delay. It then updates the resource availability of the cloudlets in the network. This process continues until all requests are assigned or no requests can be admitted by any cloudlet.

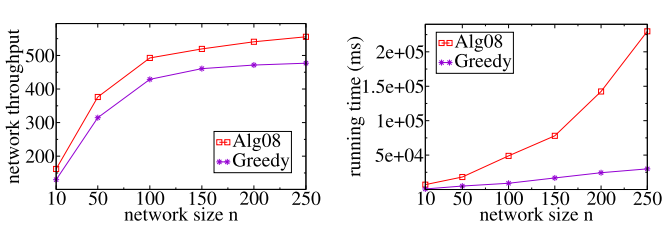
We then compare the performance of the online algorithm, `Algorithm 3` against two baseline heuristics: `OnlineNonMig` and `CBC_Greedy`, where algorithm `OnlineNonMig` does not consider VNF instance instantiations and migrations, while algorithm `CBC_Greedy` pre-deploys VNF instances by invoking algorithm `CBC` at each time frame, and makes adjustments of VNF instance provisioning by invoking heuristic `Greedy` in each time slot. For simplicity, we refer to `Algorithm 3` as algorithm `Online`.

7.2 Performance Evaluation of Different Algorithms

We start by investigating the performance of `Algorithm 1` against algorithm `CBC` for the network utility maximization problem, by varying network size from 10 to 250 while fixing the number of requests at 10,000. Fig. 3 demonstrates the total network utility gain, the number of admitted user requests, and the running time of each of the two mentioned algorithms. From Fig. 3a, it can be seen that `Algorithm 1` outperforms algorithm `CBC`, and its network utility gain is always larger than that by `CBC`. The performance gap between the two algorithms becomes larger with the increase on network size. In particular, `Algorithm 1` achieves 23.8 percent more network utility gain than that by algorithm `CBC` when the number of APs is 250. Also, it can be seen from Fig. 3b that the number of requests admitted by `Algorithm 1` is much more than that by algorithm `CBC`, and the former admits on average 965 more users than the latter. Fig. 3c depicts the running time of the two algorithms. The running time of `Algorithm 1` is a bit higher than that of algorithm `CBC` as the former examines and updates more potential VNF replicas for finding a better solution.

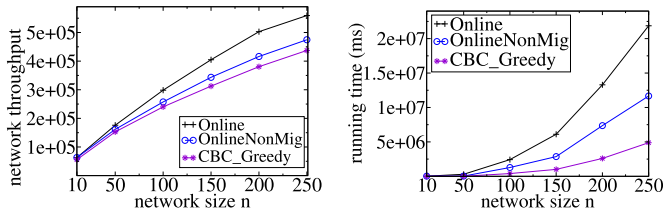
We then study the performance of `Algorithm 2` against the benchmark heuristic `Greedy` in terms of the number of requests admitted in each time slot, by varying the network size from 10 to 250 for 1,000 user requests. Fig. 4a demonstrates the number of user requests whose delay requirements can be guaranteed (network throughput) by the two algorithms. The network throughput delivered by `Algorithm 2` is better than that by algorithm `Greedy` in all cases. For example, when the network size is 200, the network throughput delivered by `Algorithm 2` is around 14.65 percent more than that by algorithm `Greedy`. With the increase on network size, the performance gap between the two algorithms becomes larger and larger. The rationale behind is that `Algorithm 2` strives for a better mapping of user requests to cloudlets, while algorithm `Greedy` only examines requests one by one and always tries to instantiate or migrate a VNF instance in its nearest cloudlet greedily. Fig. 4b plots the running time curves of the two algorithms. It can be seen that algorithm `Greedy` takes less time than that of algorithm `Algorithm 2` in all network sizes.

We finally evaluate the performance of the proposed online algorithm, `Online`, against two benchmark heuristics `OnlineNonMig` and `CBC_Greedy` for the online throughput maximization problem, by varying network size from 10 to 250 within a given time horizon that consists of 10 time frames while each time frame contains 10 time slots. Fig. 5 depicts the performance curves of different algorithms. It can be seen from Fig. 5a that algorithm `Online` outperforms its benchmark counterparts `OnlineNonMig` and `CBC_Greedy`, and the performance gap between them becomes larger with the



(a) Network throughput (b) The running time

Fig. 4. Performance of different algorithms by varying network size from 10 to 250.



(a) Network throughput (b) The running time

Fig. 5. Performance of different algorithms by varying network size from 10 to 250.

growth of network size. For example, the network throughput by Online is 8.17 and 13.23 percent more than that by algorithms OnlineNonMig and CBC_Greedy when the network size is 50, and 15.11 and 21.81 percent more than that by algorithms OnlineNonMig and CBC_Greedy when the network size is 250, respectively, from which the effectiveness of instantiating new VNF instances and migrating existing VNF replicas can be well justified. Fig. 5b depicts the running time curves of the mentioned algorithms, where CBC_Greedy takes the least running time, and OnlineNonMig takes less running time than that of algorithm Online in all network sizes, as the algorithm Online considers the pre-deployment of VNF replicas globally and employs VNF instance instantiation and migration procedure in each time slot to maximize the network throughput.

7.3 Parameter Impacts on the Algorithm Performance

We now study the impact of important parameters on the performance of the proposed algorithms including the ratio of the maximum number of sojourn locations of a user to the number of APs, the duration L of each time frame; and the length of the given time horizon. We start with analyzing the impact of the ratio on the performance of algorithms Algorithm 1 and CBC, respectively, for a set of 10,000 user requests in a network with 100 AP nodes. It can be seen from Fig. 6 that Algorithm 1 outperforms algorithm CBC. Specifically, when the ratio is 0.1, Algorithm 1 achieves 15.5 percent more network utility gains than that by algorithm CBC. Also, from Fig. 6a, it can be seen that with the increase on the ratio, users will have more dispersed sojourn locations. Thus, more computing resources are consumed for deploying VNF replicas to cope with user mobility. Similarly, the number of admitted requests decreases with the increase of the ratio. When the ratio is 0.1, on average 3,018 user requests can be admitted by Algorithm 1, while only 1,670 user requests can be admitted when the ratio is 1.0.

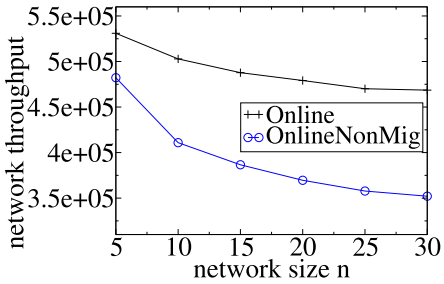


Fig. 7. The throughput performance of different algorithms by varying the length L of each time frame.

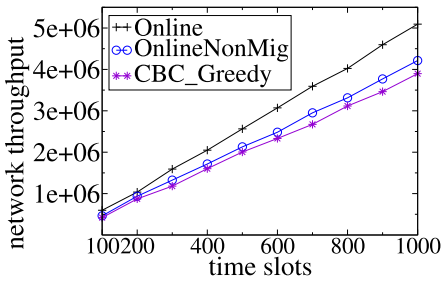


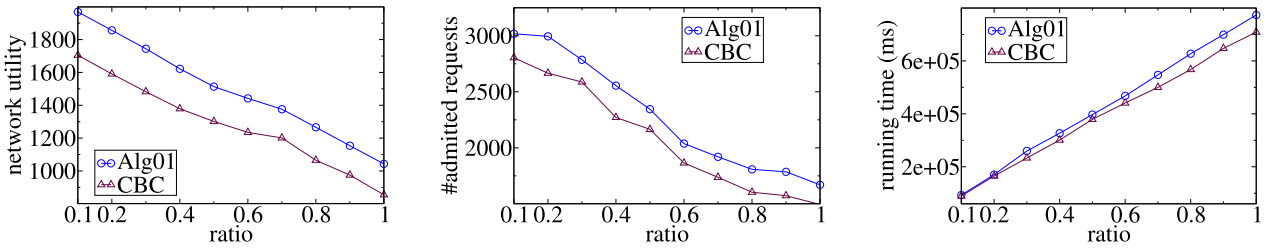
Fig. 8. The throughput Performance of different algorithms for a time horizon consisting of 1,000 time slots.

We then investigate the impact of the duration L of each time frame (the number of time slots in each time frame) on the performance of algorithms Online and OnlineNonMig in a network with 200 AP nodes. It can be seen from Fig. 7 that the performance of algorithm Online is always superior to algorithm OnlineNonMig. As algorithm Online employs the VNF migration strategy to adjust the placement of VNF instances for users, more user requests can have their delay requirements satisfied.

We also evaluate the impact of the length of time horizon on the performance of algorithm Online against the baseline heuristics OnlineNonMig and CBC_Greedy for a time horizon of 100 time frames, while fixing network size at 100. Fig. 8 shows the performance of the mentioned algorithms. From Fig. 8, it can be seen that the longer the time horizon, the larger network throughput delivered by algorithm Online, compared with those by algorithms OnlineNonMig and CBC_Greedy, respectively.

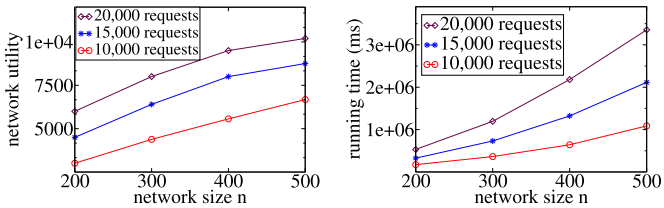
7.4 Scalability Evaluation of Different Algorithms

We finally study the scalability of the proposed algorithms in terms of performance and running time, by varying the network size from 200 to 500. Fig. 9 shows the performance of



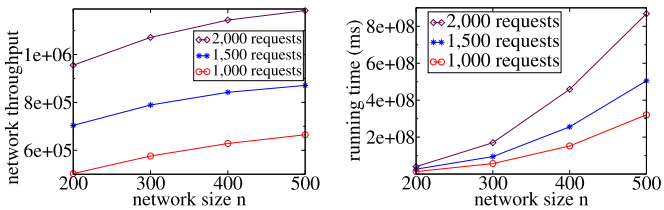
(a) The accumulative network utility gain (b) The total number of admitted requests (c) The running times

Fig. 6. Performance of different algorithms by varying the ratio of maximum number of sojourn locations to the number of access points.



(a) The accumulative network utility gain (b) The running time

Fig. 9. Performance of `Algorithm 1` by varying network size from 200 to 500, and number of requests from 10,000 to 20,000.



(a) Network throughput (b) The running time

Fig. 10. Performance of algorithm `Online` by varying network size from 200 to 500, and number of requests per time slot from 1,000 to 2,000.

`Algorithm 1` for a set of 10,000 to 20,000 requests. It can be seen from Fig. 9a that the network utility gain increases with the growth on the network size. Fig. 9b plots the running time curves of the algorithms for request sets with different sizes. It can be seen that the running time of `Algorithm 1` in large-scale networks with large numbers of requests is scalable. Fig. 10 shows the performance of algorithm `Online` for a set of 1,000 to 2,000 requests per time slot within a given time horizon that consists of 10 time frames while each time frame contains 10 time slots, the similar performance behaviors like the case for the static snapshot is observed too.

8 Conclusion

In this paper, we investigated reliable and seamless virtualized network service provisioning in a mobile edge-cloud network, by incorporating user mobility and service response delay sensitivity into consideration. We first formulated two novel optimization problems: the network utility maximization problem and the online throughput maximization problem, respectively. We then devised a constant approximation algorithm for the first problem, and developed an efficient online algorithm for the second problem. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results demonstrated that the proposed algorithms are promising.

Acknowledgments

The authors would like to appreciate the three anonymous referees and the associate editor for their constructive comments and invaluable suggestions, which help us improve the quality and presentation of the paper greatly. The work of Yu Ma, Weifa Liang, and Jing Li was supported by Australian Research Council under its Discovery Project Scheme with Grant No. DP200101985, and the work of Xiaohua Jia was supported by the Research Grants Council of Hong Kong with Project No. CityU 11214316.

References

[1] S. K. Bose, S. Brock, R. Skeoch, and S. Rao, "Cloudspider: Combining replication with scheduling for optimizing live migration of virtual machines across wide area networks," in *Proc. IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2011, pp. 13–22.

[2] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," *ACM Queue*, vol. 14, pp. 70–93, 2016.

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

[4] N. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Commun. Magazine*, vol. 47, no. 7, pp. 20–26, Jul. 2009.

[5] T. H. Cormen, C. E. Leiserson. R. L. Rivest, and C. Sten, *Introduction to Algorithms*, 3rd Ed. Cambridge, MA, USA: MIT Press, 2009.

[6] I. Farris, T. Taleb, M. Bagaa, and H. Flick, "Optimizing service replication for mobile delay-sensitive applications in 5G edge network," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[7] H. Feng, J. Llorca, A. M. Tulino, D. Raz, and A. F. Molish, "Approximation algorithms for the NFV service distribution problem," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[8] D. A. Freedman, *Statistical Models: Theory and Practice*. Cambridge, U.K.: Cambridge Univ. Press, 2009, pp. 26.

[9] GT-ITM, Oct. 2, 2019. [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

[10] X. Guan, X. Wan, J. Wang, X. Ma, and G. Bai, "Mobility aware partition of MEC regions in wireless metropolitan area networks," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2018, pp. 1–2.

[11] T. He, H. Khamfroush, S. Wang, T. L. Porta, and S. Stein, "It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources," in *Proc IEEE 38th Int. Conf. Distrib. Comput. Syst.*, 2018, pp. 365–375.

[12] 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. 4, pp. 725–737, Oct.–Dec. 2017.

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

[14] M. Jia, W. Liang, Z. Xu and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proc 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 730–738.

[15] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *J. Sel. Areas Commun.*, vol. 29, pp. 1765–1775, 2011.

[16] K. Lei, M. Qin, B. Bai, G. Zhang, and M. Yang, "GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 388–396.

[17] Y. Ma, W. Liang, M. Huang, and S. Guo, "Profit maximization of NFV-enabled request admissions in SDNs," in *Proc. IEEE Global Commun. Conf.*, 2018, pp. 1–7.

[18] Y. Ma, W. Liang, Z. Xu, and S. Guo, "Profit maximization for admitting requests with network function services in distributed clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 1143–1157, May 2019.

[19] Y. Ma, W. Liang, and S. Guo, "Mobility-aware delay-sensitive service provisioning for mobile edge computing," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2019, pp. 270–276.

[20] 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. 3, pp. 2322–2358, Fourth Quarter 2017.

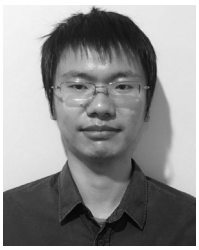
[21] J. Martins *et al.*, "ClickOS and the art of network function virtualization," in *Proc. 11th USENIX Conf. Netw. Syst. Des. Implementation*, 2014, pp. 459–473.

[22] A. Nadembega, T. Taleb, and A. Hafid, "A destination prediction model based on historical data, contextual knowledge and spatial conceptual maps," in *Proc. IEEE Int. Conf. Commun.*, 2012, pp. 1416–1420.

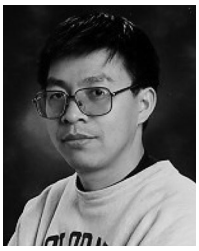
[23] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commu. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, Third Quarter 2017.

[24] T. Ojima and T. Fujii, "Resource management for mobile edge computing using user mobility prediction," in *Proc. Int. Conf. Inf. Netw.*, 2018, pp. 718–720.

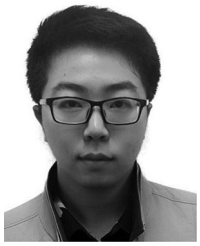
[25] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," in *Proc. IEEE/ACM 26th Int. Symp. Quality Service*, 2018, pp. 1–10.
[26] S. V. Rossem *et al.*, "Deploying elastic routing capability in an SDN/NFV-enabled environment," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, 2015, pp. 22–24.
[27] A. Santoyo-González and C. Cervello-Pastor, "Latency-aware cost optimization of the service infrastructure placement in 5G networks," *J. Netw. Comput. Appl.*, vol. 114, pp. 29–37, 2018.
[28] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5G network infrastructure," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 3879–3884.
[29] A. Vermam *et al.*, "Large-scale cluster management at Google with Borg," in *Proc. Eur. Conf. Comput. Syst.*, 2015, pp. 1–17.
[30] L. Wang, L. Jiao, J. Li, and M. Muhlhauser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 1281–1290.
[31] L. Wang, L. Jiao, T. He, J. Li, and, M. Muhlhauser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. IEEE Conf. Compute. Commun.*, 2018, pp. 468–476.
[32] S. Wang and C. Ran, "Rethinking cellular network planning and optimization," *IEEE Wireless Commun.*, vol. 23, no. 2, pp. 118–125, Apr. 2016.
[33] S. Wang, W. Zhao, and C. Wang, "Budgeted cell planning for cellular networks with small cells," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4797–4806, Oct. 2015.
[34] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Capacitated cloudlet placements in wireless metropolitan area networks," in *Proc. IEEE 40th Conf. Local Comput. Netw.*, 2015, pp. 570–578.
[35] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parallel Distribu. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2016.
[36] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao, "Task offloading with network function services in a mobile edge-cloud network," *IEEE Trans. Mobile Comput.*, vol. 18, no. 11, pp. 2072–2685, Nov. 2019.



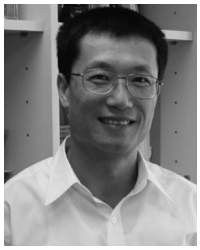
Yu Ma received the BSc degree (with the first class honors) in computer science from Australian National University, in 2014. He is currently working toward the PhD degree in the Research School of Computer Science, Australian National University. His research interests include Software Defined Networking, Internet of Things (IoT), and Social Networking.



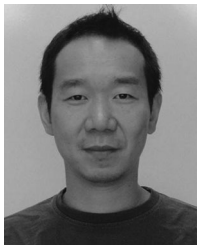
Weifa Liang (Senior Member, IEEE) received the BSc degree in computer science from Wuhan University, China, in 1984, the ME degree in computer science from the University of Science and Technology of China, in 1989, and the PhD degree in computer science from the Australian National University, in 1998. He is currently a full professor at 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, Internet of Things, mobile edge computing, Network Function Virtualization and Software-Defined Networking, design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory.



Jing Li received the BSc degree with the first class honours in computer science from Australian National University, in 2018. He is currently working toward the PhD degree in the Research School of Computer Science, Australian National University. His research interests include mobile edge computing, network function virtualization, and combinatorial optimization.



Xiaohua Jia (Fellow, IEEE) received the BSc and MEng degrees from the University of Science and Technology of China, in 1984 and 1987, respectively, and the DSc degree in information science from the University of Tokyo, in 1991. He is currently a chair professor with the Department of Computer Science, City University of Hong Kong. His research interests include cloud computing and distributed systems, computer networks, wireless sensor networks and mobile wireless networks. He is an editor of the *IEEE Transactions on Parallel and Distributed Systems* (2006-2009), the *Journal of World Wide Web*, the *Wireless Networks*, the *Journal of Combinatorial Optimization*, and so on. He is the general chair of ACM MobiHoc 2008, TPC co-chair of IEEE MASS 2009, area-chair of IEEE INFOCOM 2010, TPC co-chair of IEEE GlobeCom 2010, Ad Hoc and Sensor Networking Symposium, and Panel co-chair of IEEE INFOCOM 2011.



Song Guo (Fellow, IEEE) received the PhD degree in computer science from the University of Ottawa, and was a professor with the University of Aizu. He is a full professor with the Department of Computing, The Hong Kong Polytechnic University. His research interests include the areas of Big Data, cloud computing and networking, and distributed systems. He is the recipient of the 2017 IEEE Systems Journal Annual Best Paper Award and other five Best Paper Awards from IEEE/ACM conferences. He was an associate editor of the *IEEE Transactions on Parallel and Distributed Systems* and an IEEE ComSoc Distinguished Lecturer. He is now on the editorial board of the *IEEE Transactions on Emerging Topics in Computing*, the *IEEE Transactions on Sustainable Computing*, the *IEEE Transactions on Green Communications and Networking*, and the *IEEE Communications*. His currently serves as a director and member of the Board of Governors of ComSoc.