Maximizing User Service Satisfaction for Delay-Sensitive IoT Applications in Edge Computing

Jing Li†, Weifa Liang†, Senior Member, IEEE, Wenzheng Xu†, Member, IEEE, Zichuan Xu†, Member, IEEE, Xiaohua Jia†, Fellow, IEEE, Wanlei Zhou†, Senior Member, IEEE, and Jin Zhao†, Senior Member, IEEE

Abstract—The Internet of Things (IoT) technology provisions unprecedented opportunities to evolve the interconnection among human beings. However, the latency brought by unstable wireless networks and computation failures caused by limited resources on IoT devices prevents users from experiencing high efficiency and seamless user experience. To address these shortcomings, the integrated Mobile Edge Computing (MEC) with remote clouds is a promising platform to enable delay-sensitive service provisioning for IoT applications, where edge-clouds (cloudlets) are co-located with wireless access points in the proximity of IoT devices. Thus, computation-intensive and sensing data from IoT devices can be offloaded to the MEC network immediately for processing, and the service response latency can be significantly reduced. In this paper, we first formulate two novel optimization problems for delay-sensitive IoT applications, i.e., the total utility maximization problems under both static and dynamic offloading task request settings, with the aim to maximize the accumulative user satisfaction on the use of the services provided by the MEC, and show the NP-hardness of the defined problems. We then devise efficient approximation and online algorithms with provable performance guarantees for the problems in a special case where the bandwidth capacity constraint is negligible. We also develop efficient heuristic algorithms for the problems with the bandwidth capacity constraint. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising in reducing service delays and enhancing user satisfaction, and the proposed algorithms outperform their counterparts by at least 10.8 percent.

Index Terms—Cost modeling, resource optimization and allocation, service provisioning, delay-sensitive IoT applications, maximum profit generalized assignment problems, approximation algorithms, online algorithms, task offloading and scheduling, service delay, user satisfaction of using services, mobile edge computing (MEC)

1 INTRODUCTION

INTERNET of Things (IoT) is emerging as part of the infrastructures for advancing a large variety of applications involving the connection of intelligent devices, thereby leading to smart communities [18]. It urgently needs infrastructures and algorithms to provide effective services for delay-sensitive IoT applications, such as online gaming, augmented reality (AR), virtual reality (VR), smart cities, and autonomous vehicles. Due to the limited computing and storage resources of most IoT devices, it is common to offload computing-intensive or large storage tasks of various applications to remote clouds for processing [13]. This task offloading however suffers from a seriously high latency and network congestion in IoT infrastructures [11]. It thus is inappropriate to offload delay-sensitive IoT applications to remote clouds for processing [17]. Mobile edge computing (MEC) has emerged as a key technology to reduce network traffic, improve user experience, and enable various IoT applications [3]. The cloudlets (edge servers) are placed at the edges of core networks to provide cloud-capability services in the proximity of IoT devices and their mobile users to reduce the service response time, thereby meeting the stringent latency requirements of IoT applications.

With the fast development of 5G, MEC promises to greatly reduce the data processing delay for IoT services, by deploying computing resource (e.g., cloudlets) within the proximity of IoT devices [3]. Fuelled by the 5G technology, it is expected that the 5G-supported MEC will be the promising platform for delay-sensitive IoT services for various IoT applications. To explore the potential of MEC to support IoT applications, in this paper we deal with offloading task services in MEC for delay-sensitive IoT applications, where IoT devices are resource-constrained, by offloading their tasks to cloudlets or a remote cloud for processing. We here consider an integrated platform that consists of the remote cloud and a set of local cloudlets forming an MEC network for IoT service provisioning, where IoT devices or mobile...
users can offload their tasks to the platform for processing, and different offloading task service requests have different service delay requirements. We aim to devise efficient scheduling algorithms for assigning requests to different cloudlets or the remote cloud while meeting their service delay requirements. This poses the following challenges.

For a set of offloading task requests, which requests should be assigned to a local cloudlet and which ones should be assigned to the remote cloud for processing, considering the heterogeneity of both computing resource and processing capability of cloudlets and the remote cloud. How to assign different requests to different cloudlets or the remote cloud such that the average user experience of using the services provided by the platform is maximized while keeping the workload among all cloudlets as balanced as possible, where a user satisfaction is inversely proportional to the extra service delay beyond the user’s delay threshold; and how to develop a cost model to quantify a user satisfaction of using a service provided by the platform. In this paper, we will address the challenges and develop efficient approximation and online algorithms for delay-sensitive service provisioning for IoT applications in an integrated MEC platform.

The novelty of the work in this paper lies in that we consider the user satisfaction of using services provided by an MEC network and a remote cloud for delay-sensitive IoT applications, through maximizing the accumulative user satisfaction when different users offload their tasks with different service delay requirements. A novel metric to measure user satisfaction of using a service is proposed, and efficient approximation and online algorithms for the defined problems under both static and dynamic user service demands are then devised.

The main contributions of this paper are presented as follows.

- We consider user service satisfaction of using services provided by an MEC network and a remote cloud for delay-sensitive IoT applications, by formulating two novel user service satisfaction problems. We also show that the defined problems are NP-hard.
- We devise approximation and online algorithms with provable performance guarantees for special cases of the defined problems when the bandwidth capacity constraint is negligible. We also develop efficient heuristic algorithms for the problems with the bandwidth capacity constraint too.
- We evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.

The rest of the paper is organized as follows. Section 2 summarizes the related work on service provisioning in MEC. Section 3 introduces the system model, notions, definitions, and NP-hardness proofs of the defined problems. Section 4 devises an approximation algorithm and an efficient heuristic algorithm for the total utility maximization problem without and with the bandwidth capacity constraint, respectively. Section 5 deals with dynamic user service request admissions without the knowledge of future arrivals for a given time horizon, and efficient online algorithms for the problem are developed. Section 6 evaluates the proposed algorithms empirically, and Section 7 concludes the paper.

2 RELATED WORK

With the emergence of complicated and resource-hungry mobile applications in the Internet of Things (IoT) and smart cities, implementing user tasks on cloudlets of an MEC network becomes an important approach to shorten service response delays, reduce the energy consumption of mobile devices, and improve the user experience of using services. Task offloading in MEC networks has been extensively studied in recent years. Most existing work focused on minimizing the energy consumption of mobile devices, or the end-to-end delay of a task execution through partitioning a task into two parts: one part is offloaded to the cloudlets in the MEC for execution and another part is processed by the mobile device itself. Most task offloading concentrated on such a single task offloading.

There are extensive investigations on admitting a set of requests with the aim to minimize the average service delay of offloaded tasks. For example, Gouareb et al. [5] considered the problem of minimizing the total service delay of implementation of a service function chain in MECs, by proposing a heuristic. Huang et al. [7] considered the delay-sensitive service placement and migration in an MEC network by proposing heuristic algorithms for the problem. Jia et al. [8] studied task offloading in an MEC with the aim to minimize the average delay of all admitted requests, by incorporating queuing delays at both Access Points (APs) and cloudlets. Lyu et al. [12] investigated the joint optimization of both task admission decisions and efficient resource allocation to minimize the total energy consumption in an MEC network while the delay requirements are met. They proposed a task admission approach to achieve the asymptotic optimality by pre-admitting resource-restrained mobile devices. Xu et al. [25] considered the delay-aware service offloading problem in MEC with each request having a specific service requirement. They aimed to minimize the service cost through Virtual Network Function (VNF) instance placement, sharing, and migration, by developing online algorithms for request admissions. They however did not include the remote cloud as an alternative processing source. Xu et al. [26] also studied delay-aware service placement in MEC to minimize the operational cost, by assuming that the specified delay of each request cannot be violated, and they developed an approximation algorithm for the problem. Xia et al. [22] considered a set of delay-aware tasks to be offloaded to an MEC network with the aim to minimize the service cost. They provided a cost model, and proposed a heuristic for the problem based on the best cost model. Xia et al. [23] recently considered service caching in MEC to meet service delay requirements by proposing efficient approximation and heuristic algorithms.

Although the aforementioned investigations on delay-aware task offloading have been extensively studied in the past several years, there are only a handful of studies that take into account service provisioning in MEC platforms for delay-sensitive IoT applications. For example, Alameddine
et al. [1] studied a joint task offloading for application assignment and resource allocation in an MEC network, by formulating a dynamic task offloading and scheduling problem. They developed a mixed integer linear programming solution and a heuristic solution for the problem. Aris-dakessian et al. [2] adopted game theory and designed preference functions for IoT devices and edge nodes based on several metrics. By adopting the preference functions, they developed centralized and distributed algorithms for the assignment of IoT services to edge nodes to minimize the IoT service delay and execution makespan. Ma et al. [14], [15] studied NFV-enabled unicasting and multicasting problems with service function chain requirements, they developed heuristic and online algorithms for the problems. Samanta et al. [16] developed a dynamic microservice provisioning scheme for IoT devices in MEC environments, by formulating a novel model that incorporates both the service delay and service price into consideration. Song et al. [18] investigated the task assignment for IoT applications in an MEC network while meeting the Quality of Service (QoS) requirement of each task. They proposed a heuristic algorithm to achieve efficient network resource utilization for each microservice admission. Xu et al. [24] considered the operational cost minimization problem for implementing IoT applications with Service Function Chain (SFC) requirements, by focusing on IoT application placement in an MEC network by developing both randomized and heuristic placement algorithms. Yu et al. [27] studied the problem of IoT service provisioning with the objective to meet computing, network bandwidth and QoS requirements of each IoT application. They devised approximation algorithms to deal with the IoT service provisioning under various scenarios, respectively.

Unlike the aforementioned works focusing on either the cost minimization problem or the delay-aware service placement problem in MEC networks, in this paper we consider a set of offloading task requests from IoT devices with different service delay requirements, in which all requests must be served by either cloudlets in an MEC network or a remote cloud. We aim to maximize the accumulative user satisfaction of using the services provided by the integrated platform of the MEC network and the remote cloud. It also must be mentioned that this paper is an extension of a conference paper [10].

3 PRELIMINARIES

In this section, we first introduce the system model, we then give notions, notations, and the modeling of user service satisfaction of using services. We finally quantify the user satisfaction on a provided service in an MEC network, and define the problems precisely.

3.1 System Model

Consider a heterogeneous MEC network that is represented by an undirected graph $G = (\mathcal{AP} \cup V \cup \{v_b\}, E)$, where $\mathcal{AP}$ is the set of Access Points (APs), $V$ is the set of cloudlets, $v_b$ is the remote cloud, and $E$ is the set of links between APs. Each cloudlet (edge cloud) $v\in V$ is co-located with an AP, and connected through a high-speed optical cable, thus the communication delay between them is negligible. However, not each AP is co-located with a cloudlet, and the number of cloudlets usually is far smaller than that of APs. Each cloudlet $v\in V$ is associated with a computing capacity $C_v > 0$ and a packet processing rate $\mu_v$. Node $v_b$ is a remote cloud with unlimited computing and storage resources. Therefore, the remote cloud $v_b$ has the maximum packet processing rate, compared with cloudlets. Each link $e \in E$ has a bandwidth capacity $B(e)$. We further assume that each AP in the MEC network is connected to the remote cloud $v_b$ through a gateway in the MEC network, and the communication delay is far larger than the communication delay between any pair of APs in the MEC network.

We assume that different cloudlets have different computing resource capacities and different processing capabilities. For a given offloading task, the assignment of the task to different cloudlets will result in different computing delays as the workloads and computing capabilities at different cloudlets are different.

We consider a given time horizon that is further divided into $T$ equal time slots. Within each time slot $t$, let $\mu^v_t$ represent the processing rate of cloudlet $v\in V$, and $C^v_t$ the residual computing capacity of $v$, at time slot $t$, where $C^v_t = C^v - \mu^v_t$ for all $v\in V$, and $\mu^v_t$ is the processing capability of node $v$, which is the maximum one, i.e., $\mu^v_t = \max(\mu^v_j | 0 \leq j \leq |V|)$. We further assume that the data rate $d'_t$ for task offloading of a request $r_t$ from its nearby AP $l_t$ at time slot $t$ is fixed.

Although data uploading from an IoT device to its nearby AP is the bottleneck of some delay-sensitive applications, it becomes insignificant with the adoption of the 5G technology. Also, given a communication metric (e.g., the link congestion or the euclidean distance between the two endpoints of each physical link), let $d(e)$ be the transmission delay on a link $e$ in the MEC, which is fixed at each time slot $t$. However, the values of the mentioned parameters may change at different time slots. For the sake of convenience, we will drop index $t$ from these parameters if no confusion arises from the context.

3.2 The Service Delay of an Offloading Task for Service

Consider a set $R$ of requests, each user service request $r_t \in R$ can be expressed by a tuple $r_t = (s_t, b_t, l_t, D_t, b_t)$, where $s_t$ is the task size (volume), $b_t$ is the demanded bandwidth resource, the user of $r_t$ is under the coverage of AP $l_t$, $D_t$ is the service delay requirement threshold, and $b_t \cdot D_t$ is the maximum service delay the user could tolerate with a constant $b_t \geq 1$. Denote by $c(s_t)$ the demanded computing resource to process the offloaded task of request $r_t$. The service delay of a request consists of the uploading delay, the communication delay of routing the data from the data source to the cloudlet (or the remote cloud) for the data processing, and the processing delay of the task at the cloudlet (or the remote cloud), which are as follows.

The uploading delay $d_{upload}(l_t)$ of an offloading task $r_t$ through its located AP $l_t$ is

$$d_{upload}(r_t) = \frac{s_t}{\gamma_{l_t}},$$

where $\gamma_{l_t}$ is the uplink data rate of AP $l_t$, which can be calculated by the following Shannon-Hartley formula [20].
\[ \gamma_i = W_i \log_2(1 + \frac{k_i}{\sigma^2}), \]  

(2)

where \( W_i \) is the total bandwidth of AP \( i \), divided by the number of users under its coverage, \( k_i \) is the transmission power of IoT device of request \( r_i \), and \( \sigma^2 \) is the noise power.

An offloading task will be served by either a cloudlet or the remote cloud \( v_j \in V \cup \{v_0\} \), the communication delay \( d_{\text{comm}}(r_i, v_j) \) of offloading task \( r_i \) to node \( v_j \) is

\[
d_{\text{comm}}(r_i, v_j) = \begin{cases} \sum_{e \in \mathcal{P}(v_j)} d(e), & \text{if } v_j \in V \\ d_{\text{comm}}(r_i, v_0), & \text{otherwise } (v_j = v_0), \end{cases}
\]

(3)

where \( \mathcal{P}(v_j) \) is a routing path of request \( r_i \) between its AP location and the AP location of cloudlet \( v_j \), \( d(e) \) is the communication delay on a link \( e \), and \( d_{\text{comm}}(r_i, v_0) \) is the communication delay of routing the task of \( r_i \) from AP \( l \) to the remote cloud through the gateway.

The processing delay \( d_{\text{comp}}(r_i, v_j) \) of an offloading task \( r_i \) at cloudlet or the remote cloud \( v_j \) is

\[
d_{\text{comp}}(r_i, v_j) = \frac{s_i}{\mu_{v_j}},
\]

(4)

where \( s_i \) is the task size of request \( r_i \), and \( \mu_{v_j} \) is the processing rate of cloudlet (or the remote cloud).

The service delay \( d(r_i, v_j) \) of offloading task \( r_i \) to node \( v_j \) for service thus is defined as follows.

\[
d(r_i, v_j) = d_{\text{upload}}(r_i) + d_{\text{comm}}(r_i, v_j) + d_{\text{comp}}(r_i, v_j).
\]

Note that we do not include the delay of returning the result to the user as the result usually is no larger than the uploading volume of data, and the delay of returning the result thus is omitted.

### 3.3 User Service Satisfaction of Using a Service

In most IoT applications, each service request does have its expected delay threshold and maximum tolerable delay requirement. A task offloading request usually can be represented by a tuple \( r_i = (s_i, b_i, l_i, D_i, \beta_i) \), where \( D_i \) is its delay threshold, if the actual service delay is beyond its threshold \( D_i \), the service may still be acceptable by the user. However, in terms of the service experience, the user of request \( r_i \) may not be happy about the service. In other words, the service satisfaction of a user for his requested service can be expressed by a non-increasing function of the service delay he experienced. If a service delay is within the specified threshold, the user satisfies the service with 100 percent; otherwise, his satisfaction with the service is inversely proportional to the extra service delay beyond the user’s delay threshold.

Specifically, assume that a user request \( r_i \in R \) is assigned to cloudlet or the remote cloud \( v_j \) for service, then its service delay is \( d(r_i, v_j) \) by Eq. (5). If \( d(r_i, v_j) \) is no greater than \( D_i \), the user satisfies the service with 100 percent; otherwise, his satisfaction on the service will dramatically decrease with the increase on the value of \( d(r_i, v_j) \), and the maximum tolerable service delay of the user is \( \beta_i \cdot D_i \), where \( \beta_i \geq 1 \) is a constant, representing a certain degree of the delay tolerance of the user. If a service delay is beyond the maximum tolerable service delay of the user, the user satisfaction on the service will become zero. We thus model a user service satisfaction of using a service provided by an MEC network and a remote cloud through a non-increasing utility function as follows.

\[
u(r_i, v_j) = \begin{cases} (\lambda - \frac{[d(r_i, v_j) - D_i]^+}{\beta_i \cdot D_i}), & \text{if } d(r_i, v_j) \leq \beta_i \cdot D_i \\ 0, & \text{otherwise} \end{cases}
\]

(6)

where \([x]^+ = \max(x, 0)\), and \( \lambda > 1 \) is a constant that indicates the delay sensitivity.

It can be seen from Eq. (6) that if the service delay is no greater than \( D_i \), \( d(r_i, v_j) \leq D_i \) thus is Mb \( = 1 \), and the utility gain of the user is \( u(r_i, v_j) = \lambda - 1 \geq 0 \), implying the user is 100 percent satisfied. Otherwise, if the service delay is within the delay range of \( (D_i, \beta_i \cdot D_i) \), i.e., \( 0 < d(r_i, v_j) - D_i \leq (\beta_i - 1) \cdot D_i \) then \( \frac{[d(r_i, v_j) - D_i]^+}{\beta_i \cdot D_i} \leq \frac{\beta_i - 1}{\beta_i} < 1 \), and the utility value \( u(r_i, v_j) = \lambda - \frac{\beta_i - 1}{\beta_i} < \lambda - 1 \), i.e., the user satisfaction decreases with the growth of the delay duration and is impacted by both \( \lambda \) and \( \beta_i \). A larger value of \( \lambda \) means that the utility obtained is more sensitive than that of a smaller \( \lambda \), and a larger \( \beta_i \) implies that the user of request \( r_i \) is more tolerable to his service delay. When the actual service delay \( d(r_i, v_j) > \beta_i \cdot D_i \) that is beyond the maximum tolerable service delay of the user of \( r_i \), then \( u(r_i, v_j) = 0 \) by the utility function definition, and the user satisfaction is 0 percent. Thus, the value of \( \beta_i \) reflects the service delay tolerance of the user of request \( r_i \) at a certain extent.

### 3.4 Problem Definitions

In this paper, we consider the service provisioning in an integrated platform that consists of an MEC network and a remote cloud for delay-sensitive IoT applications, by formulating two novel optimization problems.

**Problem 1**: Given an MEC network \( G = (\mathcal{AP} \cup V \cup \{v_0\}, E) \) with a given set \( R \) of requests, each request \( r_i = (s_i, b_i, l_i, D_i, \beta_i) \) in \( R \) is expressed by a tuple, where \( s_i \) is the size of the offloading task, \( b_i \) is the demanded bandwidth resource, \( l_i \) is the physical location of the offloading task, \( D_i \) is the ideal tolerable delay threshold, and \( \beta_i \cdot D_i \) is the maximum tolerable service delay of the request. The total utility maximization problem is to maximize the utility sum of all requests in \( R \), i.e., the total user experience of using the services provided by the MEC network, subject to computing capacities on cloudlets and bandwidth capacities on links in \( G \).

As network service providers provide continuing services for their consumers, in the defined problem so far, we have only considered user requests at a given time slot \( t \), where the data rate \( y_{f}^{t} \) assigned for each user under the coverage of an AP \( l \in \mathcal{AP} \) is fixed at time slot \( t \). However, the value of \( y_{f}^{t} \) will change at a different time slot \( t^{'} \) which will be determined by the number of users under the coverage of AP \( l \) at that time slot. Meanwhile, the transmission delay \( d^{e}(l) \) on a link \( e \) in \( G \) at time slot \( t^{'} \) can also be changed, which is impacted not only by the link length but also the congestion on the link. The processing rate \( \mu_{v}^{t} \) of a
node $v \in V \cup \{v_0\}$ may vary at different time slots too. In the following we consider the dynamic user service request admissions within a finite time horizon that consists of $T$ equal time slots.

**Problem 2:** Given an MEC network $G = (\mathcal{A}P \cup V \cup \{v_0\}, E)$ and a finite time horizon that consists of $T$ equal time slots, assume that user service requests arrive one by one without the knowledge of future arrivals, the online average total utility maximization problem is to maximize the average sum of accumulative utilities of all admitted requests per time slot within the given time horizon, subject to both computing capacities on cloudlets and bandwidth capacities on links in $G$.

**Theorem 1.** The total utility maximization problem in an MEC network $G = (\mathcal{A}P \cup V \cup \{v_0\}, E)$ is NP-hard.

**Proof.** We show the claim by a reduction from a well-known NP-hard problem – the maximum profit Generalized Assignment Problem (GAP) that is defined as follows [4]. Given $n$ items and $m$ bins, if item $i$ is packed to bin $j$, it results in a profit $p_{i,j}$ and a size $s_{i,j}$. Usually the size of each item $i$ at different bins is fixed, i.e., $s_{i,j} = s_{i,j}$ even if $j \neq j'$, and each bin has a capacity. The problem is to pack as many items as possible to the $m$ bins such that the total profit of the packed items is maximized, subject to bin capacities.

We consider a special case of the total utility maximization problem where the bandwidth resource consumption of each request is negligible, by assuming that there is abundant bandwidth resource on each link in the MEC network. Thus, the routing path of routing the offloaded task of request $r_i$ to cloudlet $v$ is the routing path from AP $l_i$ to cloudlet $v$ with the least communication delay. There are $(|V|+1)$ bin, where bin $B_0$ corresponds to the remote cloud with unlimited computing capacity, and each of the other bins corresponds to a cloudlet $v \in \mathcal{C}$ with the capacity of $C_v$. Each item $i$ corresponds to a request $r_i$. For each bin $B_j$, each item has the size $s_{i,j} = c(s_i)$ and profit $p_{i,j} = u(r_i, v_j)$. The total utility maximization problem for this special case is to maximize the total utility gain by admitting as many requests as possible, subject to the computing capacities on cloudlets. It can be seen that this special problem is equivalent to the maximum profit GAP. Hence, the total utility maximization problem is NP-hard.

**4 Algorithms for the Total Utility Maximization Problem**

In this section, we deal with the total utility maximization problem. We first consider a special case of the problem where there are abundant bandwidth resources on links, for which we formulate an integer linear programming (ILP) solution for the problem when the problem size is small. Otherwise, we devise an approximation algorithm with a provable approximation ratio for the problem, by reducing the problem to the maximum profit GAP problem. An approximate solution to the latter in turn returns an approximate solution to the former. We also devise an efficient heuristic algorithm for the problem under the bandwidth capacity constraint too.

**4.1 ILP and Approximation Algorithms for the Problem Without the Bandwidth Capacity Constraint**

We deal with the total utility maximization problem without the bandwidth capacity constraint on links, by assuming that each link has abundant bandwidth resource. We start with the ILP formulation as follows.

$$\text{Maximize } \sum_{i=1}^{|R|} \sum_{j=0}^{|V|} u(r_i, v_j) \cdot x_{i,j},$$

subject to the following constraints.

$$\forall i, j, \ 1 \leq i \leq |R|, 0 \leq j \leq |V|$$

$$\sum_{j=0}^{|V|} x_{i,j} \leq 1, \ \forall i, \ 1 \leq i \leq |R|$$

$$\sum_{i=1}^{|R|} x_{i,j} \cdot c(s_i) \leq C_v, \ \forall j, \ 0 \leq j \leq |V|$$

$$x_{i,j} \in \{0, 1\}, \ \forall i, j, \ 1 \leq i \leq |R|, \ 0 \leq j \leq |V|.$$  

where variable $x_{i,j}$ is a binary variable, and $x_{i,j} = 1$ implies that offloading task $r_i$ will be served by cloudlet $v_j$ with $0 \leq j \leq |V|$. Constraint (8) ensures that each request is assigned to at most one node for service. Constraint (9) ensures that the accumulative resource demand by all requests assigned to a node is no more than the capacity of the node. Recall that we assume that the remote cloud is node $v_0$ with unlimited computing resource. Note that for each request $r_i$ in Eq. (3), its routing path $P_i(v_j)$ to cloudlet $v_j \in V$ is a shortest path in $G$ between AP $l_i$ and cloudlet $v_j$, and the weight of each link $e \in P_i(v_j)$ is the transmission delay, i.e., $d_e$, because each link is assumed to have abundant bandwidth resource.

We then devise an approximation algorithm for the problem by reducing it to the maximum profit GAP, which is a well-known NP-hard problem and there is an efficient approximation algorithm for it [4].

The reduction is as follows. There are $(|V|+1)$ bins, where bin $B_0$ corresponds to the remote cloud with unlimited computing resource, the rest $|V|$ bins correspond to the $|V|$ heterogeneous cloudlets, where $B_j$ with $1 \leq j \leq |V|$ represents cloudlet $v_j \in V$ with capacity $C_v$. There are $|R|$ requests. Recall that request $r_i \in R$ is located at AP $l_i$, if it is assigned to cloudlet $v_j$ for service with the computing resource consumption $c(s_i)$, then the utility gain is $u(r_i, v_j)$ by Eq. (6), which is determined by the experienced service delay $d(r_i, v_j)$. In other words, if we pack request $r_i$ to bin $B_j$, it generates a profit $u(r_i, v_j)$ with size $c(s_i)$, where $1 \leq j \leq |V|$; otherwise (if $r_i$ is sent to the remote cloud $v_0$ for service), its service delay is $d(r_i, v_0)$, and the utility gain is $u(r_i, v_0)$. Note that when the utility obtained by packing a request to a bin is zero, the request will not be admitted. The detailed algorithm is given in Algorithm 1.
4.2 Heuristic Algorithm for the Problem With the Bandwidth Capacity Constraint

We now consider the problem under the bandwidth capacity constraint by developing a greedy algorithm that proceeds iteratively.

Algorithm 1. An Approximation Algorithm for the Total Utility Maximization Problem Without the Bandwidth Capacity Constraint

Input: \(|V|\) cloudlets with each \(v_j \in V\) having computing capacity \(C_{v_j}\), a remote cloud \(v_0\) with unlimited computing capacity, i.e., \(C_{v_0} = \infty\), a set of requests \(R\) with each request \(r_i = (s_i, b_i, l_i, D_i, \beta_i)\).

Output: Admit as many requests as possible from \(R\) that maximizes the utility sum of admitted requests.

1: Calculate the shortest path between each cloudlet and each AP, and the weight of each link is the communication delay on the link.
2: Construct an instance of the GAP, where each request \(r_i \in R\) has a corresponding item \(i\) with size \(c(s_i)\) and the profit \(u(r_i, v_j)\). Each cloudlet \(v_j\) or the remote cloud corresponds to a bin \(B_j\) with bin capacity \(cap(B_j) = C_{v_j}\), where \(0 \leq j \leq |V|\);
3: Find an approximate solution \(A\) to the GAP problem with maximizing the utility sum, by invoking the approximation algorithm due to Cohen et al. [4];
4: for any request \(r\) with utility zero do
   5: \(A \leftarrow A \setminus \{r\}\); /* remove request \(r\) from the solution */
   6: end for;
7: return the solution \(A\) as the solution of the total utility maximization problem without the bandwidth capacity constraint.

At each iteration, for a request \(r_i \in R\) to be offloaded, we first identify the set of cloudlets \(V' \subseteq V\) and the set of links \(E' \subseteq E\) with sufficient residual computing resource and bandwidth resource to accommodate request \(r_i\), respectively. We then find a routing path \(P_i(v_j)\) in the induced subgraph \(G' = (\{AP\} \cup V' \cup \{v_0\}, E')\) of \(G = (\{AP\} \cup V \cup \{v_0\}, E)\) with the least communication delay from location \(l_i\) of request \(r_i\) to cloudlet \(v_j \in V'\), through links in \(E'\). Then, the utility gain of assigning request \(r_i\) to cloudlet \(v_j\) through the routing path \(P_i(v_j)\) can be obtained. Because the remote cloud \(v_0\) has the unlimited computing resource, the remote cloud can be identified as the offloading node of the request too. Among all nodes in \(V' \cup \{v_0\}\), we then identify a node \(v_i\) with the maximum utility gain for request \(r_i\). However, if the maximum utility gain of assigning request \(r_i\) to the node \(v_i\) is zero, the request \(r_i\) will be rejected. Among all requests to be offloaded, we identify a request \(r_x\) with the maximum utility gain for its admission. If request \(r_x\) is assigned to a cloudlet, the residual computing resource on the cloudlet and residual bandwidth resource on the links in the routing path are then updated accordingly. This procedure continues until all requests are either admitted or rejected. The detailed algorithm is given in Algorithm 2.

4.3 Algorithm Analysis

In the following we first analyze the approximation ratio and time complexity of the approximation algorithm, Algorithm 1. We then analyze the time complexity of the heuristic algorithm, Algorithm 2.

Algorithm 2. A Heuristic Algorithm for the Total Utility Maximization Problem With the Bandwidth Capacity Constraint

Input: \(|V|\) cloudlets with each \(v_j \in V\) having computing capacity \(C_{v_j}\), a remote cloud \(v_0\) with unlimited computing capacity, i.e., \(C_{v_0} = \infty\), each link \(e \in E\) connecting cloudlets has a bandwidth capacity, and a set of requests \(R\) with each request \(r_i = (s_i, b_i, l_i, D_i, \beta_i)\).

Output: Admit as many requests as possible from \(R\) that maximizes the utility sum of admitted requests.

1: \(R \leftarrow R\); /* the requests to be offloaded */
2: \(A \leftarrow \emptyset\); /* the solution */
3: while \(R \neq \emptyset\) do
   4: for each request \(r_i \in R\) do
      5: Identify the set of cloudlets \(V' \subseteq V\) and the set of links \(E' \subseteq E\) with sufficient residual resource for \(r_i\);
      6: Find the routing path \(P_i(v_j)\) from \(AP\) \(l_i\) to each cloudlet \(v_j \in V'\) with the least communication delay, through links in \(E'\);
      7: Calculate the utility gain \(u(r_i, v_j)\) if \(r_i\) is assigned to each cloudlet \(v_j \in V'\) through the routing path \(P_i(v_j)\);
      8: Calculate the utility gain \(u(r_i, v_0)\) if \(r_i\) is assigned to the remote cloud \(v_0\);
      9: Find node \(v_i \in V' \cup \{v_0\}\) with the maximum utility gain \(u(r_i, v_i)\) for request \(r_i\);
     10: if \(u(r_i, v_i) > 0\) then
        11: \(r_i\) is rejected;
        12: \(R \leftarrow R \setminus \{r_i\}\);
     13: end if;
   14: end for;
   15: Find request \(r_x\) with the maximum utility gain, and admit request \(r_x\) by assigning request \(r_x\) to node \(v_j\);
   16: \(A \leftarrow A \cup \{r_x\}\); \(R \leftarrow R \setminus \{r_x\}\);
   17: if \(v_j\) is a cloudlet then
      18: Update the residual resource on cloudlet \(v_j\) and the links on the routing path \(P_i(v_j)\);
   19: end if;
   20: end while;
21: return the solution \(A\) as the solution of the total utility maximization problem with the bandwidth capacity constraint;

Lemma 1. Given an MEC network \(G = (\{AP\} \cup V \cup \{v_0\}, E)\) and a set \(R\) of user requests, the upper bound on the optimal solution of the total utility maximization problem in \(G\) is \((\lambda - 1) \cdot |R|\).

Proof. The claim of that the optimal solution is upper bounded by \((\lambda - 1) \cdot |R|\) is shown as follows. If a request \(r_i \in R\) can be served within its specified delay threshold, i.e., \(d(r_i, v_j) \leq D_i\), the utility obtained by this service is \((\lambda - 1)\); if \(D_i < d(r_i, v_j) \leq \beta_i \cdot D_i\), its utility is \(\lambda - \frac{\lambda - 1}{\beta_i} < \lambda - 1\); otherwise, its utility is 0. \(\square\)

Theorem 2. Given an MEC network \(G = (\{AP\} \cup V \cup \{v_0\}, E)\) and a set \(R\) of offloading task requests, there is an approximation algorithm, Algorithm 1, for the total utility maximization problem without the bandwidth capacity constraint, which delivers an approximate solution with a \(\frac{1}{3}\) approximation
ratio. The time complexity of the approximation algorithm is $O(|V| \cdot |AP|^2 + |V| + 1 \cdot |R| \cdot \log \frac{1}{\epsilon} + \frac{|V|+1}{\epsilon^2})$, where $\epsilon$ is a constant with $0 < \epsilon \leq 1$.

**Proof.** The approximation ratio of the proposed algorithm, Algorithm 1 can be obtained by adopting the analysis of the approximation algorithm due to Cohen et al. [4]. The solution delivered by the algorithm is no less than $\frac{1}{1+\epsilon}$ times the optimal one, where $\epsilon$ is a constant with $0 < \epsilon \leq 1$.

The running time of Algorithm 1 is analyzed as follows. Finding the shortest paths between each cloudlet and each AP takes $O(|V| \cdot |AP|^2)$ time, while the approximation algorithm due to Cohen et al. [4] takes $O((|V| + 1) \cdot |R| \cdot \log \frac{1}{\epsilon} + \frac{|V|+1}{\epsilon^2})$ time. Thus, the time complexity of Algorithm 1 is $O(|V| \cdot |AP|^2 + (|V| + 1) \cdot |R| \cdot \log \frac{1}{\epsilon} + \frac{|V|+1}{\epsilon^2})$.

**Theorem 3.** Given an MEC network $G = (AP \cup V \cup \{v_0\}, E)$ and a set $R$ of offloading task requests, there is an algorithm, Algorithm 2, for the total utility maximization problem with the bandwidth capacity constraint, which delivers a feasible solution, and its time complexity is $O(|V|^2 \cdot |AP|^2)$.

**Proof.** It can be seen that the solution delivered by Algorithm 2 is feasible because no specified constraint is violated. The time complexity of Algorithm 2 is analyzed as follows.

There are at most $|R|$ iterations. Within each iteration, a request with the maximum utility gain from the remaining requests is admitted. The time of calculating the utility gain of admitting a request $r_i$ is dominated by the time of finding the shortest path from the location of request $r_i$ to each cloudlet with sufficient computing resource through links with sufficient bandwidth resource for $r_i$, which takes $O(|AP|^2)$ time. Thus, the time complexity of the proposed algorithm is $O(|V|^2 \cdot |AP|^2)$.

## 5 Online Algorithms for the Online Average Total Utility Maximization Problem

In this section, we study dynamic user service request admissions, where user service requests arrive one by one without the knowledge of future arrivals, and all arrived requests will be considered in the beginning of the next time slot. We start with a special case of the problem where the bandwidth capacity constraint is not considered, for which we devise an online algorithm with a provable competitive ratio. We then develop an efficient online algorithm for the problem with the bandwidth capacity constraint.

Notice that once an admitted request finishes its service, its occupied resources will be released back to the system in the end of the time slot it leaves. Thus, the available capacity of each cloudlet or link at each time slot is its residual capacity at that time slot, and those occupied resources are not available for new request admissions at the next time slot.

### 5.1 Online Algorithm for the Problem Without the Bandwidth Capacity Constraint

Denote by $C_v(i)$ the residual computing resource at cloudlet $v \in V$ before considering request $r_i$, and $C_v(1) = C_v$ initially.

If request $r_i$ is assigned to cloudlet $v$, we can calculate the utility gain if request $r_i$ is assigned to cloudlet $v$ through the shortest routing path. Thus, the available computing resource to accommodate request $r_i$, i.e., $v$ is assigned to cloudlet $v$, will be rejected if both

$$w_v(i) = C_v \left( \alpha \frac{C_v(i)}{C_v} - 1 \right),$$

where $\alpha > 1$ is a tuning parameter reflecting the sensitivity of the workload at each cloudlet $v_i$ and $1 - \frac{C_v(i)}{C_v}$ is the utilization ratio of cloudlet $v_i$.

The normalized computing resource cost of assigning offloading request $r_i$ to cloudlet $v$ thus is

$$\psi_v(i) = \frac{w_v(i)}{C_v} = \alpha \left( \frac{C_v(i)}{C_v} - 1 \right) \left( \frac{C_v}{C_v(i)} - 1 \right).$$

Upon the arrival of request $r_i$, we first identify the set $V' \subseteq V$ of cloudlets with sufficient residual computing resource to accommodate request $r_i$. Then, we find the shortest routing path from the located AP $l_i$ of request $r_i$ to each cloudlet $v \in V'$ and calculate the utility gain if request $r_i$ is assigned to cloudlet $v$ through the shortest routing path. Among all cloudlets in $V'$, we identify the set of cloudlets $Q_i \subseteq V'$ with the positive utility gains for request $r_i$, and request $r_i$ is assigned to the cloudlet in $Q_i$ with the minimum normalized computing resource cost by Eq. (12). If no such cloudlet exists, request $r_i$ can then be assigned to the remote cloud with unlimited computing resource. However, if the utility gain brought by assigning request $r_i$ to the assigned node is $0$ (i.e., $d(r_i, v_j) > \beta_i \cdot D_i$), the request can be rejected.

We now assume that request $r_i$ is assigned to node $v_j' \in V \cup \{v_0\}$ with the utility gain $v_i$. If request $r_i$ is assigned to the remote cloud (i.e., $v_j = v_0$) with a positive utility gain, it indicates that the request will be admissible. Although $r_i$ is admissible, its admission needs further examination to avoid consuming too much resource, by adopting an admission control policy. That is, request $r_i$ will be rejected if both the following conditions are met. (i) The normalized computing resource cost of cloudlet $v_j' \in V'$ that will accommodate request $r_i$ is greater than $\frac{|V| \cdot v_i}{v_j' \cdot u_j}$, i.e., $\psi_v(i) > \frac{|V| \cdot v_i}{v_j' \cdot u_j}$; and (ii) assigning request $r_i$ to the remote cloud will result in the zero utility gain (i.e., exceeding the maximum tolerable service delay). Note that if condition (i) is met while condition (ii) is violated (i.e., assigning request $r_i$ to the remote cloud will result in a positive utility gain), request $r_i$ is admitted and assigned to the remote cloud.

The detailed online algorithm with a provable competitive ratio is given in Algorithm 3.

### 5.2 Online Algorithm for the Problem With the Bandwidth Capacity Constraint

We then deal with the online average total utility maximization problem with the bandwidth capacity constraint by devising an efficient online algorithm as follows.

Recall that for the problem without the bandwidth capacity constraint, we introduce a computing resource cost...
model to capture the dynamic consumptions of computing resources on cloudlets. Similarly, we here introduce the bandwidth resource cost model to capture the dynamic bandwidth resource consumptions of links as follows.

\[ w_e(i) = B_e(\delta - \frac{B_e(i)}{B_e} - 1), \]

where \( \delta > 1 \) is a turning parameter reflecting the sensitivity of the workload at each link \( e \). \( B_e \) is the bandwidth capacity of link \( e \in E \), \( B_e(i) \) is the residual bandwidth resource on link \( e \in E \) before considering request \( r_i \), and \( 1 - \frac{B_e(i)}{B_e} \) is the utilization ratio of link \( e \).

The normalized bandwidth cost of link \( e \in E \) for request \( r_i \) thus is

\[ \psi_e(i) = \frac{w_e(i)}{B_e} = \delta - \frac{B_e(i)}{B_e} - 1. \]

The normalized bandwidth cost on a routing path \( P_v(v) \) of request \( r_i \) then is \( \sum_{e \in P_v(v)} \psi_e(i) \).

The total normalized cost of assigning request \( r_i \) to cloudlet \( v \) through the routing path \( P_v(v) \) consists of the normalized computing resource cost \( \psi_v(i) \) on \( v \) and the normalized bandwidth cost on \( P_v(v) \), i.e.,

\[ \phi(P_v(v)) = \psi_v(i) + \sum_{e \in P_v(v)} \psi_e(i). \]

Similar to Algorithm 3, the proposed online algorithm proceeds as follows. Upon the arrival of request \( r_i \), we first identify the set of cloudlets \( V^e \subseteq V \) and the set of links \( E^e \subseteq E \) with sufficient computing and bandwidth resources to accommodate request \( r_i \). We then assign each cloudlet \( v \in V^e \) with a normalized computing resource cost \( \psi_v(i) \) by Eq. (12) and a normalized bandwidth resource cost \( \psi_e(i) \) by Eq. (14), respectively. We find a routing path \( P_v(v) \) in the induced subgraph \( G' = (AP \cup V' \cup \{v_i\}, E') \) of graph \( G = (AP \cup V \cup \{v_i\}, E) \) with the least communication delay on the path from location \( l_i \) of request \( r_i \) to each cloudlet \( v_j \in V^e \). Hereafter, among all cloudlets in \( V^e \), we finally identify the set of cloudlets \( Q_i \subseteq V^e \) with the positive utility gains for request \( r_i \) through the associated routing paths. Then, among all cloudlets in \( Q_i \), we assign request \( r_i \) to cloudlet \( v' \in Q_i \) through its routing path with the minimum total normalized cost by Eq. (15). If no such a cloudlet exists, the request can be assigned to the remote cloud. Note that if condition (i) is met while condition (ii) is violated (i.e., assigning request \( r_i \) to the remote cloud will result in a positive utility gain), request \( r_i \) is admitted and assigned to the remote cloud.

**Algorithm 3.** Online Algorithm for the Online Average Total Utility Maximization Problem Without the Bandwidth Capacity Constraint

**Input:** \( |V| \) cloudlets with each \( v_j \in V \) having computing capacity \( C_{v_j} \), a remote cloud \( v_0 \) with unlimited computing capacity, i.e., \( C_{v_0} = \infty \), a set of requests \( R \) with each request \( r_i = (s_i, b_i, l_i, D_i, \beta_i) \) arrived one by one, there is no knowledge of future request arrivals.

**Output:** Maximize the average total utility gain of admitted requests per time slot within the time horizon.

1. \( A = \emptyset; /* the solution */ \)
2. while request \( r_i \) arrives do
3. \( Q_i = \emptyset; /* the set of candidate cloudlets for \( r_i \) */ \)
4. if both the following conditions are met: (i) The normalized computing resource cost of cloudlet \( v' \) or the normalized bandwidth resource cost of the routing path \( P_v(v') \) is greater than \( |V| \cdot u_i \), i.e., \( \psi_v(i) > |V| \cdot u_i \), or \( \sum_{e \in P_v(v')} \psi_e(i) > |V| \cdot u_i \); and (ii) the utility gain is zero if the request is assigned to the remote cloud.
5. Assign \( r_i \) to remote cloud;
6. update the residual computing resource of cloudlet \( v' \);
7. if its utility gain is positive then
8. \( Q_i \leftarrow Q_i \cup \{v_j\}; \)
9. end if;
10. end for;
11. if \( Q_i = \emptyset \) then
12. Assign \( r_i \) to remote cloud makes positive utility gain;
13. if assigning \( r_i \) to remote cloud makes positive utility
14. then
15. Assign \( r_i \) to remote cloud;
16. end if;
17. else
18. Reject \( r_i \);
19. if assigning \( r_i \) to remote cloud makes positive utility then
20. Assign \( r_i \) to remote cloud;
21. if assigning \( r_i \) to remote cloud makes positive utility then
22. Assign \( r_i \) to remote cloud;
23. else
24. Reject \( r_i \);
25. end if;
26. else
27. Assign \( r_i \) to remote cloud;
28. Update the residual computing resource of cloudlet \( v' \);
29. end if;
30. end if;
31. if \( r_i \) is admitted then
32. \( A = A \cup \{r_i\}; \)
33. end if;
34. end while;
35. return a feasible solution \( A \) to the online average total utility maximization problem without the bandwidth capacity constraint;

The detailed algorithm is given in Algorithm 4.
Algorithm 4. A Heuristic Algorithm for the Online Average Total Utility Maximization Problem With the Bandwidth Capacity Constraint

Input: $|V|$ cloudlets with each $v_j \in V$ having computing capacity $C_{v_j}$, a remote cloud $v_0$ with unlimited computing capacity, i.e., $C_{v_0} = \infty$, each link $e \in E$ connecting cloudlets has a bandwidth capacity, and a set of requests $R$ with each request $r_i = \langle s_i, t_i, l_i, d_i, \beta_i \rangle$ arrived one by one, there is no knowledge of future request arrivals.

Output: Maximize the average total utility gain of admitted requests per time slot within the time horizon.

1: $A \leftarrow \emptyset$; /* the solution */
2: while request $r_i$ arrives do
3: Identify the set of cloudlets $V' \subseteq V$ and the set of links $E' \subseteq E$ with sufficient residual resource for $r_i$;
4: Find the routing path $P_i(v_j)$ from AP to each cloudlet $v_j \in V'$ with the least communication delay, through links in $E'$;
5: $Q_i \leftarrow \emptyset$; /* the set of candidate cloudlets for $r_i$ */
6: for each cloudlet $v_j \in V'$ do
7: if the utility gain $u(r_i, v_j)$ of assigning $r_i$ to node $v_j$ through $P_i(v_j)$ is positive then
8: $Q_i \leftarrow Q_i \cup \{v_j\}$;
9: end if;
10: end for;
11: if $Q_i = \emptyset$ then
12: if assigning $r_i$ to remote cloud makes positive utility then
13: Admit $r_i$ by assigning $r_i$ to remote cloud;
14: else
15: Reject $r_i$;
16: end if;
17: else
18: Identify the cloudlet $v' \in Q_i$ and its routing path $P_i(v')$ with the minimum total normalized cost by Eq. (15). And calculate the utility gain $u_i$ by assigning $r_i$ to $v'$ through $P_i(v')$;
19: if $\psi_i(i) > |V| \cdot u_i$ OR $\sum_{v \in P_i(v')} \psi_v(i) > |V| \cdot u_i$ then
20: if assigning $r_i$ to remote cloud makes positive utility gain then
21: Admit $r_i$ by assigning $r_i$ to remote cloud;
22: else
23: Reject $r_i$;
24: end if;
25: else
26: Admit $r_i$ by assigning $r_i$ to cloudlet $v'$ through $P_i(v')$;
27: Update the residual resource of $v'$ and links on $P_i(v')$;
28: end if;
29: end if;
30: if $r_i$ is admitted then
31: $A \leftarrow A \cup \{r_i\}$;
32: end if;
33: end while;
34: return a feasible solution $A$ to the online average total utility maximization problem with the bandwidth capacity constraint;

5.3 Algorithm Analysis

The rest is to analyze the competitive ratio and time complexity of Algorithm 3. The time complexity of Algorithm 4 is also analyzed.

Let $R$ be the set of requests arrived for the given time horizon. Denote by $Z(i) \subseteq R$ the set of requests admitted by Algorithm 3 prior to the arrival of request $r_i$, and $u_{\text{max}}$ and $u_{\text{min}}$ the maximum and minimum utility gains of admitting any request, respectively. Following Eq. (6), for a request $r_i$, $u_{\text{max}} = \lambda - 1$ when $d(r_i, v_i) \leq D_i$, while $u_{\text{min}} = \min_{r \in R} (\lambda - \beta_i \cdot r_i)$. When $d(r_i, v_i) = \beta_i \cdot D_i$, and both $u_{\text{max}}$ and $u_{\text{min}}$ are constants.

Lemma 2. Given an MEC network $G = (\mathcal{A} \cup V \cup \{v_0\}, E)$ and a finite time horizon that consists of $T$ time slots, let $R$ be the set of requests arriving one by one within the given time horizon, denote by $Z(i)$ the set of requests admitted by Algorithm 3 prior to the arrival of request $r_i$. Then, the sum of usage cost of all cloudlets is

$$
\sum_{v \in V} \psi_v(i) \leq 2 \cdot |V| \cdot \log(\alpha) \cdot \sum_{r \in Z(i)} (c(s_r) \cdot u_r),
$$

where $\alpha$ is a constant with $2|V| \cdot u_{\text{max}} + 2 \leq \alpha \leq 2|V| \cdot u_{\text{max}}$, $u_{\text{max}} = \lambda - 1$, $C_{\text{min}} = \min\{C_v \mid v \in V\}$, and $C_{\text{max}} = \max\{C_v \mid r_i \in R\}$.

The proof of Lemma 2 is contained in Section 1 of the supplementary materials file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2021.3107137.

Denote by $D(i)$ the set of requests admitted by the optimal solution but rejected by Algorithm 3 prior to the arrival of request $r_i$, and denote by $\mathcal{H}(i)$ the set of requests admitted by both the optimal solution and Algorithm 3 prior to the arrival of request $r_i$. It can be seen that set $D(i) \cup \mathcal{H}(i)$ is the set of admitted requests by the optimal solution. Then, for each request $r_i \in \mathcal{H}(i)$, we have

$$u_i^* \leq \frac{u_{\text{max}}}{u_{\text{min}}} \cdot u_i,
$$

where $u_i^*$ and $u_i$ are the utility gains of admitting request $r_i$ by the optimal solution and Algorithm 3, respectively, while $u_{\text{max}}$ and $u_{\text{min}}$ are the maximum and minimum utilities by admitting any request, which are constants.

Lemma 3. Given an MEC network $G = (\mathcal{A} \cup V \cup \{v_0\}, E)$ and a finite time horizon that consists of $T$ time slots, let $R$ be the set of requests arriving one by one over the time horizon, denote by $D(i)$ the set of requests admitted by the optimal solution but rejected by Algorithm 3 prior to the arrival of request $r_i$, denote by $V_{\text{eff}}$ the node in the optimal solution to which request $r_{\text{eff}} \in D(i)$ is assigned. We have $V_{\text{eff}} \in V$, $\forall r_{\text{eff}} \in D(i)$, i.e., the requests in the set $D(i)$ are assigned to cloudlets instead of the remote cloud in the optimal solution.

Proof. We prove the lemma by contradiction. We assume that there is a request $r_{\text{eff}} \in D(i)$ that is assigned to the remote cloud in the optimal solution. It can be seen that if a positive utility gain can be obtained when $r_{\text{eff}}$ is assigned to the remote cloud, then $r_{\text{eff}}$ can be assigned to the remote cloud by Algorithm 3. However, request $r_{\text{eff}}$ is rejected by Algorithm 3. This results in a contradiction. □

Lemma 4. Given an MEC network $G = (\mathcal{A} \cup V \cup \{v_0\}, E)$ and a finite time horizon that consists of $T$ time slots, let $R$ be
the set of requests arriving one by one over the time horizon, denote by $D(i)$ the set of requests admitted by the optimal solution but rejected by Algorithm 3 prior to the arrival of request $r_i$. Denote by $v^*_r$ the node in the optimal solution to which request $r_i \in D(i)$ is assigned, and denote by $u^*_r$ the utility for request $r_i \in D(i)$ in the optimal solution. Then, for each request $r_i \in D(i)$, we have

$$\psi_{v^*_r}(i') > |V| \cdot \frac{u_{\text{min}}}{u_{\text{max}}} \cdot u^*_r,$$

when $2|V| \cdot u_{\text{max}} + 2 \leq \alpha \leq 2\frac{\text{C}_{\text{max}}}{\text{C}_{\text{min}}}$. The proof of Lemma 4 is contained in Section 2 of the supplementary materials file, available online.

**Lemma 5.** Given an MEC network $G = (\mathcal{AP} \cup \mathcal{V} \cup \{\mathcal{E}\}, \mathcal{E})$ and a finite time horizon that consists of $T$ time slots, let $R$ be the set of requests arriving one by one over the time horizon, denote by $D(i)$ the set of requests admitted by the optimal solution but rejected by Algorithm 3 prior to the arrival of request $r_i$. Denote by $\mathcal{P}_{\text{opt}}(i)$ and $\mathcal{P}(i)$ the total utilities of admitted requests by an optimal solution and the solution delivered by Algorithm 3 prior to the arrival of request $r_i$, respectively. We have

$$\mathcal{P}_{\text{opt}}(i) \leq \frac{u_{\text{max}}}{u_{\text{min}}} \cdot \mathcal{P}(i) + \sum_{r_i \in D(i)} u^*_r.$$

**Proof.** Recall that $D(i) \cup \mathcal{H}(i)$ is the set of admitted requests by the optimal solution. We have

$$\mathcal{P}_{\text{opt}}(i) = \sum_{r_i \in \mathcal{H}(i)} u^*_r + \sum_{r_i \in D(i)} u^*_r \leq \frac{u_{\text{max}}}{u_{\text{min}}} \cdot \sum_{r_i \in \mathcal{H}(i)} u^*_r + \sum_{r_i \in D(i)} u^*_r,$$

by Eq. (17)

$$\leq \frac{u_{\text{max}}}{u_{\text{min}}} \cdot \mathcal{P}(i) + \sum_{r_i \in D(i)} u^*_r.$$

(20)

Ineq. (20) holds because $\mathcal{H}(i)$ is the subset of admitted requests by Algorithm 3. \qed

**Theorem 4.** Given an MEC network $G = (\mathcal{AP} \cup \mathcal{V} \cup \{\mathcal{E}\}, \mathcal{E})$ and a finite time horizon that consists of $T$ time slots, let $R$ be the set of requests arriving one by one over the time horizon, there is an online algorithm with a competitive ratio of $O(\log |V|)$, Algorithm 3, for the online average total utility maximization problem without the bandwidth capacity constraint, which takes $O(\text{AP}^2)$ time to admit each request when $\alpha = 2|V| \cdot u_{\text{max}} + 2$, where $u_{\text{max}}$ is the maximum utility gain of a request.

The proof of Theorem 4 is contained in Section 3 of the supplementary materials file, available online.

**Theorem 5.** Given an MEC network $G = (\mathcal{AP} \cup \mathcal{V} \cup \{\mathcal{E}\}, \mathcal{E})$ and a finite time horizon that consists of $T$ time slots, let $R$ be the set of requests arriving one by one over the time horizon, there is an online algorithm, Algorithm 4, for the online average total utility maximization problem with the bandwidth capacity constraint.

**Proof.** It can be seen that the solution delivered by Algorithm 4 is feasible because all constraints imposed on the problem are met.

The time complexity of Algorithm 4 for a request admission is dominated by the time of finding the routing paths with the least communication delay from AP $l_i$ to cloudlets with sufficient resource for request $r_i$, which takes $O(\text{AP}^2)$ time. \qed

**6 PERFORMANCE EVALUATION**

In this section we conduct the performance evaluation on the proposed algorithms. We also investigate the impact of important parameters on the performance of the proposed algorithms.

**6.1 Environment Setting**

We consider a heterogeneous MEC network consisting of 200 APs, 10 percent of which are co-located with cloudlets [9], [26]. The topologies of MEC networks are generated by GT-ITM [6]. For each AP, the bandwidth at each time slot is drawn from 20 MHz to 40 MHz randomly [19], the signal-to-noise ratio (i.e., $\gamma^2$) of an AP is set as 30 dB [21]. For each cloudlet, the capacity varies from 3,000 MHz to 7,000 MHz [25] and its processing rate varies from 0.5 MB to 2 MB per ms [15]. For each request, its task size is randomly drawn from 1 MB to 5 MB [18], the demanded computing resource is randomly drawn from 20 MHz to 300 MHz [26] and the demanded bandwidth resource ranges from 5 Mbps to 50 Mbps [27]. The delay requirement threshold of a request is randomly drawn from 10 ms to 50 ms [15], and $\beta$, ranges from 1 to 3. The bandwidth capacity of each link varies from 200 Mbps to 2,000 Mbps [15], and the transmission delay of a link at each time slot is chosen from 2 ms to 5 ms randomly [25], while the transmission delay from an AP to the remote cloud through the gateway varies from 80 ms to 100 ms. We further assume the processing rate of the remote cloud is 20 MB per ms. Parameter $\lambda$ is set as 2 and parameter $\epsilon$ is set as 0.5. The turning parameters $\alpha$ and $\delta$ are set as $2|V| \cdot u_{\text{max}} + 2$, where $|V|$ is the number of cloudlets and $u_{\text{max}} = \lambda - 1$. We assume that there are 100 time slots and 1000 requests arrive at each time slot one by one. The duration of each request is randomly drawn from 1 to 3 time slots [15]. The result in each figure is the mean of the results by applying an algorithm on 20 MEC network instances of the same size, where the running time of an algorithm is the actual amount of time to find a solution, based on a desktop with a 3.60 GHz Intel 8-Core i7-7700 CPU and 16 GB RAM. Unless specified, the above parameters will be adopted in the default setting.

To evaluate Algorithm 1 (referred to as Alg.1) for the total utility maximization problem without the bandwidth capacity constraint, we propose two comparison benchmarks. One is the ILP solution (7) (referred to as Optimal) which is the optimal solution to the problem; another is a greedy algorithm (referred to as Optimal), which picks requests in $R$ randomly, and assigns the picked request to the cloudlet (or the remote cloud) with the maximum utility gain, this procedure continues until all requests are assigned. To study Algorithm 2 (referred to as Alg.2) for the total utility maximization problem with the bandwidth capacity constraint, ...
capacity constraint, we also give a comparison algorithm for it, which is a greedy algorithm (referred to as Gdy.2) that requests are picked randomly. For each picked request, it first finds a routing path with the least communication delay from the location of the request to each cloudlet with sufficient computing resource, through the links with sufficient bandwidth resource for the request. It then assigns the request to the cloudlet (or the remote cloud) with the maximum utility gain. This procedure continues until all requests are assigned.

To investigate the performance of Algorithm 3 (referred to as Alg.3) for the online average total utility maximization problem without the bandwidth capacity constraint, a comparison online algorithm (referred to as Gdy.3) is proposed, which is the online version of Gdy.1. To evaluate Algorithm 4 (referred to as Alg.4) for the online average total utility maximization problem with the bandwidth capacity constraint, a greedy algorithm (referred to as Gdy.4), which is the online version of Gdy.2, is also proposed for the performance evaluation purpose.

6.2 Performance Evaluation of the Proposed Algorithms for the Total Utility Maximization Problem

We first studied the performance of Alg.1 against algorithms Optimal and Gdy.1, by varying the number of requests from 100 to 1,000. We then evaluated the performance of Alg.2 against algorithm Gdy.2, by varying the number of requests from 100 to 1,000. Figs. 1 and 2 depict the accumulated utilities and running times of different algorithms for the total utility maximization problem without and with the bandwidth capacity constraint. It can be seen from Fig. 1a that when the number of requests reaches 1,000, the performance achieved by algorithm Gdy.1 is 88.5 percent of that by Alg.1 while the performance achieved by Alg.1 is 85.2 percent of that by algorithm Optimal. Meanwhile, it can be seen from Fig. 2a that Alg.2 outperforms algorithm Gdy.2 on the performance improvement by at least 10.8 percent with 1,000 requests. The rationale behind is that both Alg.1 and Alg.2 better utilize the network resource by provisioning satisfying services to more users, compared with the greedy algorithms, and they take much less running time in comparison with the ILP solution that takes a much longer running time.

We then studied the impact of network size on the proposed algorithms with 1,000 requests, by varying the number of network size from 50 to 250. Recall that 10 percent of APs are co-located with cloudlets. Figs. 3a and 3b demonstrate the accumulated utilities by different algorithms for the total utility maximization problem without and with the bandwidth capacity constraint. We can see from Fig. 3a that when the network size is 250, the performance achieved by algorithm Gdy.1 is 76.3 percent of that by Alg.1 while the performance achieved by Alg.1 is 84.8 percent of that by Optimal. The similar performance behaviors can be observed in Fig. 3b too. This is because both Alg.1 and Alg.2 facilitate the efficient cooperation between the remote cloud and local cloudlets to maximize the accumulated user satisfaction when the network size is large.

6.3 Performance Evaluation of the Proposed Algorithms for the Online Average Total Utility Maximization Problem

We first studied the performance of Alg.3 and Alg.4 against algorithms Gdy.3 and Gdy.4, respectively, by varying the number of requests arriving at each time slot from 100 to 1,000, over the time horizon (100 time slots). Figs. 4 and 5 plot the average utilities and running times of different algorithms for the online average total utility maximization problem without and with the bandwidth capacity constraint. With 1,000 requests arriving at each time slot, in Fig. 4a algorithm Alg.3 outperforms Gdy.3 by 22.1 percent, while in Fig. 2a algorithm Alg.4 outperforms Gdy.4 by 16.4 percent. This can be justified by that either Alg.3 or Alg.4 establishes an efficient admission control policy to admit requests with larger utility gain but less resource consumption, without any knowledge of future request arrivals.

The rest is to investigate the impact of important parameters on the performance of the proposed algorithms including parameters $\beta$, $\alpha$, and $\delta$, where parameter $\beta$ reflects the service delay tolerance of user $u_i$, parameters $\alpha$ and $\delta$ reflect...
the sensitivity of the workload at each cloudlet and link, respectively. We also study the performance of the online algorithms with and without adopting the admission control policy. For the sake of convenience, in the rest experimental simulations, it is assumed that the time horizon consists of 100 time slots and 1,000 requests arrive at each time one by one.

We first evaluated the impact of parameter $b_i$ on the performance of the proposed algorithms, by varying the network size from 50 to 250. Fig. 6 illustrates the impact of parameter $b_i$ on the proposed algorithms $\text{Alg}_3$ and $\text{Alg}_4$ when $b_i = 1, 2, 3$ respectively. It can be seen from Fig. 6a that when the network size is 50, the performance of $\text{Alg}_3$ with $b_i = 1$ is 24.8 percent of itself with $b_i = 3$. When the network size is 250, the performance of $\text{Alg}_3$ with $b_i = 1$ is 76.2 percent of itself with $b_i = 3$. The similar performance behavior can be found in Fig. 6b. The rationale behind is that a larger $b_i$ leads to a larger tolerable service delay and more requests can be admitted. In addition, when the network size is small (i.e., the available computing resource is very limited), the mobile users have to better utilize the remote cloud to process their requests that result in longer service delays. Thus, a larger $b_i$ is important in admitting requests when the network size is small.

We then studied the impact of parameter $a$ on the performance of the algorithm $\text{Alg}_3$, and the impacts of parameter $a$ and $d$ on the performance of the algorithm $\text{Alg}_4$, by varying the network size from 50 to 250. Fig. 7a demonstrates the performance of $\text{Alg}_3$ with parameter $a = 2|V| \cdot u_{\text{max}} + 2, 4|V| \cdot u_{\text{max}} + 2$, and $8|V| \cdot u_{\text{max}} + 2$, respectively, where $|V|$ is the number of cloudlets, and $u_{\text{max}} = \lambda - 1$ is the maximum possible utility gain for a request. While Fig. 7b shows the performance of $\text{Alg}_4$ with parameter $\alpha = \delta = 2|V| \cdot u_{\text{max}} + 2, 4|V| \cdot u_{\text{max}} + 2$, and $8|V| \cdot u_{\text{max}} + 2$, respectively. As depicted by Fig. 7a, when the network size is 250, the performance of $\text{Alg}_3$ with $a = 8|V| \cdot u_{\text{max}} + 2$ is 89.3 percent of itself with $a = 2|V| \cdot u_{\text{max}} + 2$. The similar performance behavior can be found in Fig. 7b. The justification is that with a larger $\alpha$ or $\delta$, the normalized cost of computing resource or bandwidth resource becomes higher by Eqs. (12) and (14), and it intends to be conservative and reject requests.

Finally, we investigated the impact of the admission control policy, by varying the network size from 50 to 250. Figs. 8a and 8b plot the performance of algorithms $\text{Alg}_3$ and $\text{Alg}_4$ with and without the admission control policy. It can be seen from Fig. 8a that when the network size is 250, the performance of $\text{Alg}_3$ without the admission control policy is 86.9 percent of itself with the admission control policy. This performance behavior can be found in Fig. 8b. This can be justified by that with a reasonable admission control policy, the requests with larger utility gains but less computing resource consumption will be admitted. Thus,

Fig. 4. Performance of different algorithms for the online average total utility maximization problem without the bandwidth capacity constraint.

Fig. 5. Performance of different algorithms for the online average total utility maximization problem with the bandwidth capacity constraint.

Fig. 6. The impact of $b_i$ on the performance of the proposed algorithms.

Fig. 7. The impacts of the parameter $a$ and $d$ on the performance of the proposed algorithms.

Fig. 8. The impacts of the admission control policy on the performance of the proposed algorithms.
the admission control policy is important to deal with the dynamic request admissions.

7 Conclusion

In this paper, we studied the user satisfaction on the use of services for delay-sensitive IoT applications in an edge computing environment, by offloading user service requests to either the remote cloud or local cloudlets in an MEC network. We first formulated two novel optimization problems and showed their NP-hardness. We then proposed efficient approximation and heuristic algorithms for the admissions of a set of requests. We also developed online algorithms for the admissions of dynamic requests without the knowledge of future arrivals. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.

Acknowledgments

The authors appreciate the three anonymous referees and the Associate Editor for their constructive comments and invaluable suggestions, which have helped us improve the quality and presentation of the paper significantly. The work by Jing Li and Weifa Liang was partially supported by Australian Research Council under its Discovery Program under Grant DP200101985. The work by Wenzheng Xu was supported by the National Natural Science Foundation of China (NSFC) under Grant 61602330, Sichuan Science and Technology Program under Grants 2018GZX0010 and 2017GZX0003, and the National Key Research and Development Program of China under Grant 2017YFB0202403. The work of Zichuan Xu was partially supported by the National Natural Science Foundation of China under Grant 61802048 and the Xinghai Scholar Program in Dalian University of Technology, China. The work by Xiaohua Jia was supported by the Research Grants Council of Hong Kong with Project No. CityU 11214316.

References

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

Weifa Liang (Senior Member, IEEE) received the BSc degree in computer science from Wuhan University, China, in 1984 and the ME and PhD degrees in computer science from Australian National University, in 1989 and 1998, respectively. He is currently a professor with the Department of Computer Science, City University of Hong Kong. Prior to that, he was a professor with Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, mobile edge computing (MEC), network function virtualization (NFV), Internet of Things, software-defined networking (SDN), design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is currently an associate editor for the editorial board of IEEE Transactions on Communications.

Wenzheng Xu (Member, IEEE) received the BSc, ME, and PhD degrees in computer science from Sun Yat-Sen University, Guangzhou, China, in 2008, 2010, and 2015, respectively. He is currently an associate professor with Sichuan University, China. He was a visitor with Australian National University, Australia, and the Chinese University of Hong Kong, Hong Kong. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory.

Zichuan Xu (Member, IEEE) received the BSc and ME degrees in computer science from the Dalian University of Technology, China, in 2008 and 2011, respectively, and the PhD degree from Australian National University, Australia, in 2016. From 2016 to 2017, he was a research associate with the Department of Electronic and Electrical Engineering, University College London, U.K. He is currently an associate professor and PhD advisor with the School of Software, Dalian University of Technology, China. He is also a Xinghai scholar with the Dalian University of Technology, China. His research interests include cloud computing, mobile edge computing, deep learning, network function virtualization, software-defined networking, routing protocol design for wireless networks, algorithmic game theory, and optimization problems.

Jin Zhao (Senior Member, IEEE) received the BE degree in computer science from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2001 and the PhD degree in computer science from Nanjing University, Nanjing, in 2006. He is currently an associate professor with Fudan University, Shanghai, China. In 2014, he was a visiting scholar with the University of Massachusetts Amherst, Amherst, MA, USA, for one year. His research interests include software-defined networking and network function virtualization.

Xiaohua Jia (Fellow, IEEE) received the BSc and ME 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. From 2006 to 2009, he was an editor of the IEEE Transactions on Parallel and Distributed Systems, Journal of World Wide Web, Wireless Networks, and Journal of Combinatorial Optimization. He is the general chair of ACM MobiHoc 2008, the TPC co-chair of IEEE MASS 2009, the area-chair of IEEE INFOCOM 2010, the TPC co-chair of IEEE GlobeCom 2010, Ad Hoc and Sensor Networking Symposium, and the panel co-chair of IEEE INFOCOM 2011.

Wanlei Zhou (Senior Member, IEEE) received the BE and ME degrees in computer science and engineering from the Harbin Institute of Technology, Harbin, China, in 1982 and 1984, respectively, and the PhD degree in computer science and engineering from the Australian National University, Canberra, Australia. He is currently the vice director (academic affairs) and dean of the Institute of Data Science, City University of Macau, Macao, China. His research interests include distributed systems, network security, and privacy preservation.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.