Maximizing the Quality of User Experience of Using Services in Edge Computing for Delay-Sensitive IoT Applications

Jing Li
Australian National University
Canberra, ACT, Australia

Weifa Liang
Australian National University
Canberra, ACT, Australia

Wenzheng Xu
Sichuan University
Chengdu, P.R. China

Zichuan Xu
Dalian University of Technology
Dalian, P.R. China

Jin Zhao
Fudan University
Shanghai, P.R. China

ABSTRACT

The Internet of Things (IoT) technology offers unprecedented opportunities to interconnect 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 MEC with remote clouds is a promising platform, where edge-clouds (cloudlet) are co-located with wireless access points in the proximity of IoT devices, thus intensive-computation and sensing data from IoT devices can be offloaded to the MEC network for processing, and the service response latency can be significantly reduced. In this paper, we study delay-sensitive service provisioning in an MEC network for IoT applications. We first formulate two novel optimization problems, 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 of using the services provided by the MEC. We then show that the defined problems are NP-hard. We instead devise efficient approximation and online algorithms with provable performance guarantees for the problems. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.

KEYWORDS

Cost modeling; heterogeneous MEC networks; service provisioning; delay-sensitive services; generalized assignment problems; user service experience; approximation algorithms; online algorithms; the average service delay; the user experience of using services; edge computing.

ACM Reference Format:
Jing Li, Weifa Liang, Wenzheng Xu, Zichuan Xu, and Jin Zhao. 2020. Maximizing the Quality of User Experience of Using Services in Edge Computing for Delay-Sensitive IoT Applications. In *23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '20), November 16–20, 2020, Alicante, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3416010.3423234

1 INTRODUCTION

Internet of Things (IoT) is emerging as part of the infrastructures for advancing a large variety of applications involving connection of many intelligent devices, leading to smart communities. It urgently needs infrastructures and algorithms to provide effective services for delay-sensitive IoT applications, such as online gaming, augmented reality (AR) and virtual reality (VR), smart cities and smart homes, autonomous vehicles, and so on. Due to 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. This task offloading however suffers a seriously high latency and network congestion in IoT infrastructures. It thus is inappropriate to offload delay-sensitive IoT applications to remote clouds for processing [8]. Mobile edge computing (MEC) has emerged as a key technology to reduce network traffic, improve user experience, and enable various IoT applications [1], where edge servers (or cloudlets) are placed at the edge of core networks that can 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, mobile edge computing (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 [1]. 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 edge servers or a remote cloud for processing. We here consider an integrated platform that consists of the remote cloud and a set of local edge servers 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 allocating requests to different edge servers or the remote cloud while meeting their service delay requirements. This poses the following challenges.

For a set of offloading task requests, which should be allocated to a local edge server or the remote cloud for processing, considering the heterogeneity of both computing resource and processing capability of edge servers; how to allocate different requests to different edge servers or the remote cloudlet such that the average user experience of using the services provided by the platform is maximized, 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 the integrated MEC platform.

The novelty of the work in this paper lies in that we consider the user experience of using services provided by an integrated platform consisting of an MEC network and a remote cloud, for delay-sensitive IoT applications, through maximizing the accumulative user experience if different offloading tasks have different service delay requirements. We develop a novel metric to measure a user satisfaction of using a service through the service latency the user experienced, and devise efficient approximation and online algorithms for the defined problems under static and dynamic offloading task requests settings.

The main contributions of this paper are presented as follows. We consider service provisioning in an MEC network for delay-sensitive IoT applications by formulating two novel user experience satisfaction problems. We first show that the defined problems are NP-hard. We then devise efficient approximation and online algorithms with provable performance guarantees for the defined problems, under both static and dynamic admissions of offloading task requests. We finally 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 notions, and notations Section 4 defines the problems mathematically and shows NP-hardness of the defined problems. Section 5 devises an approximation algorithm for the total utility maximization problem. Section 6 shows how to extend the proposed algorithm for dynamic admissions of offloading task requests within a finite time horizon, without the knowledge of future request arrivals. Section 7 evaluates the proposed algorithms empirically, and Section 8 concludes the paper.

2 RELATED WORK

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 also quite a few investigations on admitting a set of requests with the aim to minimize the average service delay of offloaded tasks. For example, Xia [12] considered a set of delay-aware tasks to be offloaded to an MEC network with the aim to minimize the service cost of offloaded tasks and proposed a heuristic for the problem. Jia *et al.* [4] are the first to study task offloading in an MEC with the aim to minimize the average delay of all admitted requests, by incorporating queuing delays at both APs and cloudlets. They [5, 6] later considered the average delay minimization problem in an MEC, through balancing the workload among cloudlets.

Although the above studies on the 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, Song *et al.* [8] considered the QoS-based task allocation in MEC for IoT applications, by proposing efficient algorithms. Yu *et al.* [17] studied the problem of IoT service provisioning with the objective to meet computing, network bandwidth and QoS requirements of each IoT application. Xu *et al.* [16] studied the QoS-aware VNF placement of service function chains in MEC for IoT applications. Xu *et al.* [15] also considered the operational cost minimization problem for implementing IoT applications with SFC requirements, by focusing on IoT application placement in an MEC network through proposing randomized and heuristic placement algorithms.

Unlike the aforementioned work 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 local cloudlets (edge servers) or the remote cloud, we aim to maximize the accumulative user satisfaction of using the services provided by an integrated platform consisting of the MEC network and a remote cloud.

3 PRELIMINARIES

In this section, we first introduce the system model, notions and notations. We then quantify the user satisfaction on a service provided by the MEC platform.

3.1 System model

Consider that a heterogeneous MEC network is represented as an undirected graph $G = (V \cup \{v_0\}, E)$, where V is the set of nodes and E is the set of links between nodes. An AP may or may not be co-located with a cloudlet (edge cloud). The cloudlet $v \in V$ has different computing capacity $C_v > 0$ and packet processing rate μ_v, and the AP and its co-located cloudlet are connected through a high-speed optical cable, and the communication delay between them thus is negligible. Node v_0 is a remote cloud with unlimited computing and communication resources while the communication delay between any AP and node v_0 is no less than the communication delay between any two AP nodes in the MEC. And the task processing delay at the node v_0 is the fastest one due to that there are many servers in the cloud.

We consider a given time horizon that is further divided into T equal time slots. Within each time slot t, let $\mu^t_{v_j}$ represent the processing rate of cloudlet $v_j \in V$, and $C'^t_{v_j}$ the residual computing capacity of v_j at time slot t, where $C'^1_v = C_v$ for all $v \in V$, and $\mu^t_{v_0}$ is the processing capability of node v_0, which is the maximum one, i.e., $\mu^t_0 = \max\{\mu^t_j \mid 1 \le j \le |V|\}$. We further assume that the bandwidth $B^t_{v_j}$ for data uploading of a request from an AP $v_i \in V$ at time slot t is fixed, which is the total bandwidth of AP v_i is divided by the number of users after its coverage.

Given a communication metric (e.g., the link congestion or the Euclidean distance between the two endpoints of each physical link), let $l^t(v_i, v_j)$ be the length of the shortest path (delay) between two APs 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

We admit task offloading service requests at the beginning of each time slot. Consider a set R of requests, each user service request $r_i \in R$ can be expressed by a tuple $r_i = \langle s_i, l_i, D_i, \beta_i \rangle$, where s_i is the task size (volume) and l_i is the AP location of the task or the user of r_i is under the coverage of AP v_{l_i}, D_i is the service delay requirement and $\beta_i \cdot D_i$ is the maximum service delay the user could tolerate with a constant $\beta_i \geq 1$. *The service delay of a request* consists of the uploading delay, the communication delay of routing the data between the data source and the cloudlet or the remote cloud for the data processing, and the processing delay of the task at the cloudlet, which are as follows.

The uploading delay of an offloading task is

$$d_{upload}(r_i, v_{l_i}) = \frac{s_i}{B_{v_{l_i}}}, \tag{1}$$

where $B_{v_{l_i}}$ is the uplink date rate of AP v_{l_i} , which can be calculated by the following Shannon-Hartley formula [10].

$$B_{v_{l_i}} = W_{v_{l_i}} \log_2(1 + \frac{P_i}{R^{\alpha}}), \tag{2}$$

where $W_{v_{l_i}}$ is the total bandwidth of AP v_{l_i}, P_i is the transmission power of IoT device of request r_i, and α is the path loss exponent with a value range between 2 and 6, and we set it at 2.

If the offloading task will be served by a cloudlet or the remote cloud v_j with $0 \leq j \leq |V|$, then *the communication delay of an offloading task* is

$$d_{comm}(v_{l_i}, v_j) = l(v_{l_i}, v_j), \tag{3}$$

where $l(v_a, v_b)$ is the length of a shortest path in G between nodes v_a and v_b and the weight of each link in G is a metric such as the Euclidean distance between the two endpoints or the utilization ratio of the workload to its bandwidth capacity, which is determined by the service provider.

The processing delay of an offloading task at cloudlet or the remote cloud v_j is

$$d_{comp}(r_i, v_j) = \frac{s_i}{\mu_j}, \tag{4}$$

where s_i is the task size (volume) , and μ_j is the processing rate of cloudlet (or the remote cloud) at that time slot.

The service delay of offloading task r_i to cloudlet v_j for service thus is defined as follows.

$$d(r_i, v_{l_i}, v_j) = d_{upload}(r_i, v_{l_i}) + d_{comm}(v_{l_i}, v_j) + d_{comp}(r_i, v_j), \tag{5}$$

where v_{l_i} is the nearby AP of request r_i and its offloading task will be offloaded to a cloudlet or the remote cloud via the AP. Note that we do not include the delay of the processing result to the user as the result usually is no larger than the uploading volume of data, the delay of the returning the result is no more than $d(r_i, v_{l_i}, v_j)$ and thus is omitted.

3.3 User satisfaction of using a service

In most IoT application scenarios, each service request does have its expected delay threshold and maximum tolerable delay requirement. If the actual service delay $d(r_i, v_{l_i}, v_j)$ is no greater than the delay requirement D_i, the user satisfies the service with 100%; otherwise, his satisfaction on the service will dramatically decrease with the increase on $d(r_i, v_{l_i}, v_j)$, and the maximum tolerant service delay of the user is $\beta_i \cdot D_i$, where $\beta_i \geq 1$ is a constant, representing a certain degree of delay tolerance of the user. If a service delay is beyond the maximum tolerant service delay of the user, the user satisfaction on the service will become zero. We thus model a user experience of using a service provided by MEC through *a non-increasing utility function* as follows.

$$u(r_i, v_j) = \begin{cases} (\lambda - \lambda^{\frac{[d(r_i, v_{l_i}, v_j) - D_i]^+}{\beta_i \cdot D_i}}), & \text{if } d(r_i, v_{l_i}, v_j) \leq \beta_i \cdot D_i \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

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, $[d(r_i, v_{l_i}, v_j) - D_i]^+ = 0$, then $\lambda^0 = 1$, and the utility value of $u(r_i, v_j) = \lambda - 1$. The user of r_i is 100% satisfied; otherwise, if the service delay is within the delay range of $(D_i, \beta_i \cdot D_i]$, i.e., $0 < d(r_i, v_{l_i}, v_j) - D_i \leq (\beta_i - 1) \cdot D_i$, then $\frac{[d(r_i, v_{l_i}, v_j) - D_i]^+}{\beta_i \cdot D_i} = \frac{d(r_i, v_{l_i}, v_j) - D_i}{\beta_i \cdot D_i} \leq \frac{(\beta_i - 1) D_i}{\beta_i \cdot D_i} = \frac{\beta_i - 1}{\beta_i} < 1$, and the utility value $u(r_i, v_j) = \lambda - \lambda^{\gamma} < \lambda - 1$ with $0 < \gamma \leq \frac{\beta_i - 1}{\beta_i} < 1$, i.e., the user satisfaction decreases with the growth of the delay duration and is impacted by both λ and β_i. A larger value of λ means that the utility obtained is more sensitive than that of a smaller λ, and a larger β_i implies that users are more tolerable to their service delays. When the actual service delay $d(r_i, v_{l_i}, v_j) > \beta_i \cdot D_i$, it is beyond the maximum tolerant service delay of the user, then $u(r_i, v_j) = 0$, and the user satisfaction is 0%. Thus, the value of β_i reflects the service delay tolerance of a user at a certain extent.

4 PROBLEM FORMULATIONS

In this section, we consider delay-sensitive service provisioning in an MEC network for IoT applications by defining two novel optimization problems and formulating ILP solutions to them.

Problem 1: We assume that the cloudlets (edge -servers) in an MEC are heterogeneous, different cloudlets have different amounts of computing resource and processing capacities. This implies that the processing delay of the same offloading task service at different cloudlets may be different. Given a set $R = \{r_1, r_2, \ldots, r_{|R|}\}$ of service requests, if request $r_i \in R$ is sent to the remote cloud v_0 for processing, its transmission delay $d_{comm}(v_{l_i}, v_0)$ is far larger than $\max\{d_{comm}(v_i, v_j) \mid 1 \leq i, j \leq |V|\}$, and the service processing delay $d_{comp}(r_i, v_0)$ is no greater than the minimum one $\min\{d_{comp}(r_i, v_j) \mid 1 \leq j \leq |V|\}$, where $r_i = \langle s_i, l_i, D_i, \beta_i \rangle$. *The total utility maximization problem* in an MEC network $G(V \cup \{v_0\}, E)$ with a given set R of requests is to maximize the utility sum of all requests in R, where each request r_i consists of the size s_i of the offloading task, its physical location AP_{l_i}, a tolerable delay threshold D_i. The problem is to maximize the total user experience of using the services provided by the MEC network, i.e., the optimization objective is to maximize the sum of the utilities of all requests.

$$\text{Maximize} \sum_{i=1}^{|R|}\sum_{j=0}^{|V|} u(r_i, v_j) \cdot x_{i,j}, \tag{7}$$

subject to the following constraints.

$$Eq.\ (1),\ (3),\ (4),\ (5),\ (6),\ \forall i,j\ 1 \le i \le |R|,\ 0 \le j \le |V| \tag{8}$$

$$\sum_{j=0}^{|V|} x_{i,j} = 1,\ \forall i\ 1 \le i \le |R| \tag{9}$$

$$\sum_{i=1}^{|R|} x_{i,j} \cdot c(s_i) \le C_{v_j},\ \forall j\ 0 \le j \le |V| \tag{10}$$

$$x_{i,j} \in 0, 1,\ \forall i,j\ 1 \le i \le |R|,\ 0 \le j \le |V|, \tag{11}$$

where $x_{i,j}$ is a binary variable, where $x_{i,j} = 1$ implies that offloading task r_i will be served by cloudlet/the remote cloud v_j with $0 \le j \le |V|$. Constraint (9) ensures that each request is assigned to only one cloudlet for service. Constraint (10) ensures that the accumulative resource demand by all requests assigned to a cloudlet is no more than the capacity of the cloudlet.

In the following, we consider dynamic service provisioning in MEC for delay-sensitive IoT applications within a given time horizon that consists of T equal time slots, where the requests arrive one by one without the knowledge of future requests.

Problem 2: Given a time horizon T and an MEC network $G(V \cup \{v_0\}, E)$, offloading task 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 during the T time slots, assuming that each request has a service delay requirement and the maximum tolerable service delay constant $\beta_i \ge 1$, i.e., our optimization objective is to

$$\text{maximize } \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{|R^t|}\sum_{j=0}^{|V|} u^t(r_i, v_j) \cdot x_{i,j,t}, \tag{12}$$

subject to the following constraints.

$$Eq.\ (1),\ (3),\ (4),\ (5),\ (6),$$
$$\forall i,j,t\ 1 \le i \le |R^t|,\ 0 \le j \le |V|,\ 1 \le t \le T \tag{13}$$

$$\sum_{j=0}^{|V|} x_{i,j,t} = 1,\ \forall i,t\ 1 \le i \le |R^t|,\ 1 \le t \le T \tag{14}$$

$$\sum_{i=1}^{|R|} x_{i,j,t} \cdot c(s_i) \le C'^t_{v_j},\ \forall j,t\ 0 \le j \le |V|,\ 1 \le t \le T \tag{15}$$

$$x_{i,j,t} \in 0, 1,\ \forall i,j,t\ 1 \le i \le |R|,\ 0 \le j \le |V|,\ 1 \le t \le T. \tag{16}$$

where $C'^t_{v_j}$ is the residual computing capacity of cloudlet $v_j \in V$ at time slot t and R^t is the set of requests arrived at time slot t with $1 \le t \le T$ and $C'^1_v = C_v$ for all $v \in V$.

Notice that the ILP formulation (12) is an offline version of the online average total utility maximization problem, assuming that all arrived requests at each time slot t are known in advance. In fact, the online version assumes that there is no knowledge of request arrivals in the future.

Theorem 4.1. *The total utility maximization problems are NP-hard.*

Proof The problem can be reduced from the maximum profit GAP [2], while the GAP with profit maximization is NP-hard, the defined problems are NP-hard, too. ■

5 APPROXIMATION ALGORITHM FOR THE TOTAL UTILITY MAXIMIZATION PROBLEM

In this section we deal with the total utility maximization problem by reducing the problem to the maximum utility GAP problem, and an approximate solution to the latter in turn returns an approximate solution to the former.

The maximum utility GAP is defined as follows. Given n items and m bins, if item i is packed to bin j, it results in a profit $p_{i,j}$ and its 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 $j \ne j'$. Each bin j has a capacity, the problem is to pack as many items as possible to the m bins such that the total profit of packed items is maximized, subject to bin capacities. This is a well known NP-complete problem, and there is an efficient approximation algorithm for it [2].

5.1 Approximation algorithm

The reduction is as follows. There are $n + 1$ bins, where bin $\mathcal{B}_0$ corresponds the remote cloud with unlimited computing resource, the rest of n bins correspond the $|V|$ heterogeneous cloudlets, where $\mathcal{B}_i$ with $1 \le i \le |V|$ represents cloudlet $v_i \in V$ with computing capacity C_{v_i}. There are $|R|$ requests. Recall that request $r_i \in R$ is located at AP v_{l_i}, if it is assigned to cloudlet v_j for service, the utility obtained is $u(r_i, v_j)$ by Eq. (6), which is determined by the service delay $d(r_i, v_{l_i}, v_j)$, and the computing resource consumption $c(s_i)$. In other words, if we pack request r_i to bin $\mathcal{B}_j$, it generates a profit $u(r_i, v_j)$ with size $c(s_i)$, where $1 \le j \le |V|$; otherwise (if r_i is sent to the remote cloud v_0 for service), its service delay is $d(r_i, v_{l_i}, v_0) = d_{upload}(r_i, v_{l_i}) + d_{comm}(v_{l_i}, v_0) + d_{comp}(r_i, v_0)$, and the profit obtained is $u(r_i, v_0)$. Note that when the utility obtained by packing a request to a bin is zero, the request may not be admitted. The detailed algorithm is given in `Algorithm` 1.

Algorithm 1 An approximation algorithm for the total utility maximization problem

```
Require: $|V|$ cloudlets with each $v_i \in V$ having computing capacity $C_{v_i}$, a remote
    cloud $v_0$ with unlimited computing capacity, i.e., $C_{v_0} = \infty$, a set of requests $R$
    with each request $r_i = \langle s_i, l_i, D_i, \beta_i \rangle$ where $1 \le i \le |R|$.
Ensure: Admit as many requests as possible from $R$ that maximizes the utility sum
    of admitted requests.
1: Construct an instance of the GAP, where each request $r_i \in R$ has a corresponding
   item $i$ with size $c(s_i)$ and each cloudlet $v_j$ or the remote cloud corresponds a bin
   $\mathcal{B}_j$ with bin capacity $cap(\mathcal{B}_j) = C_{v_j}$, where $0 \le j \le |V| + 1$. If request $r_i$ is
   assigned to cloudlet or the remote cloud $v_j$ for service, the profit obtained is the
   utiltiy $u(r_i, v_j)$ by Eq. (6);
2: Find an approximate solution $A$ to the GAP problem with maximizing the utility
   sum, by invoking the approximation algorithm due to Cohen et al. [2];
3: for any request $r \in A$ with utility zero do
4:     $A \leftarrow A \setminus \{r\}$; /* remove request $r$ from the solution */;
5: end for;
6: return the set $A$ of admitted requests as the solution to the problem.
```

5.2 Algorithm analysis

In the following we analyze the approximation ratio and time complexity of the approximation algorithm, `Algorithm` 1.

Lemma 5.1. *Given an MEC network $G = (V \cup \{v_0\}, E)$ 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 that the optimal solution is upper bounded by $(\lambda - 1) \cdot |R|$ is quite trivial, as if a request can be served within its specified delay threshold, the utility obtained by this service is $(\lambda - 1)$; otherwise, its utility is $\lambda - \lambda^{\epsilon} < \lambda - 1$ with $0 < \epsilon \le 1$. ■

Theorem 5.2. *Given an MEC network $G = (V \cup \{v_0\}, E)$ and a set R of offloading task requests from IoT devices or mobile users, there is an approximation algorithm for the total utility maximization problem,* `Algorithm` *1, which delivers an approximate solution with a $\frac{1}{2+\epsilon}$ approximation ratio. The time complexity of the approximation algorithm is $O(\frac{|R|\cdot(|V|+1)}{\epsilon} + \frac{|V|+1}{\epsilon^4})$, where ϵ is a constant with $0 < \epsilon \le 1$.*

Proof The approximation ratio of the proposed algorithm can be obtained directly by applying the approximation algorithm due to Cohen *et al.* [2], omitted.

We here analyze the running time of `Algorithm` 1. The construction of the GAP instance takes $O(|R| \cdot (|V| + 1))$ time, while the approximation algorithm due to Cohen *et al.* [2] takes $O(\frac{|R|\cdot(|V|+1)}{\epsilon} + \frac{|V|+1}{\epsilon^4})$ time. The solution delivered by the proposed algorithm, `Algorithm` 1, thus is no less than $\frac{1}{2+\epsilon}$ times the optimal one, where ϵ is a constant with $0 < \epsilon \le 1$. ■

6 ONLINE ALGORITHM FOR THE ONLINE AVERAGE TOTAL UTILITY MAXIMIZATION PROBLEM

In this section, we study the online version of the total utility maximization problem, where all arrived requests will be considered at the beginning of the next time slot, and there is no knowledge of future request arrivals. We devise an online algorithm with a provable competitive ratio for the problem, by adopting an admission control policy.

6.1 Online algorithm

The proposed online algorithm for the problem achieves a provable competitive ratio, through adopting an admission control policy. We also analyze the competitive ratio of the proposed algorithm.

Denote by $C_v(i)$ the residual computing resource at cloudlet $v \in V$ before considering request r_i. If request r_i is allocated to cloudlet v for service, $C_v(i+1) = C_v(i) - c(s_i)$. Otherwise, request r_i is allocated to the remote cloud for service, and nothing will be done because the remote cloud has unlimited resource. To capture the computing resource usage in cloudlets, a resource usage cost model is introduced as follows.

$$w_v(i) = C_v(\alpha^{1-\frac{C_v(i)}{C_v}} - 1), \tag{17}$$

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

The normalized cost of assigning offloading task r_i to cloudlet v thus is

$$\psi_v(i) = \frac{w_v(i)}{C_v} = \alpha^{1-\frac{C_v(i)}{C_v}} - 1 \tag{18}$$

Upon the arrival of request r_i, among all cloudlets with sufficient residual computing resource and positive utility gain, request r_i is assigned to the cloudlet with the minimum normalized cost by Eq. (18). If no such cloudlet exists, request r_i can then be allocated to the remote cloud with unlimited computing resource. However, if the utility gain brought by allocating request r_i to the assigned node is 0 (i.e., $d(r_i, v_{l_i}, v_j) > \beta_i \cdot D_i$), the request can be rejected.

We now assume that request r_i is assigned to node $v \in V \cup \{v_0\}$ with the utility gain u_i. If request r_i is assigned to the remote cloud (i.e., $v = v_0$) with a positive utility gain, it will be admitted. Although r_i is admissible with the utility gain u_i when it is assigned to cloudlet $v \in V$, its admission needs further examined, by adopting the following admission control policy. Request r_i will be rejected if both the following conditions are met. (i) The normalized cost of cloudlet $v \in V$ that will accommodate request r_i is greater than $|V| \cdot u_i$, i.e., $\psi_v(i) > |V| \cdot u_i$; and (ii) allocating 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., allocating 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` 2.

6.2 Algorithm analysis

The rest is to analyze the competitive ratio of `Algorithm 2` as follows.

Denote by $\mathcal{Z}(i) \subseteq R$ the set of requests admitted by `Algorithm` 2 prior to the arrival of request r_i. Denote by u_{max} and u_{min} the maximum and minimum utility gains of admitting any request, respectively. Following Eq. (6), for a request r_i, $u_{max} = \lambda - 1$ when $d(r_i, v_{l_i}, v_j) \le D_i$, while $u_{min} = \min_{r_i \in R}\{\lambda - \lambda^{\frac{\beta_i - 1}{\beta_i}}\}$ when $d(r_i, v_{l_i}, v_j) = \beta_i \cdot D_i$, where both u_{max} and u_{min} are constants.

Lemma 6.1. *Given an MEC network $G = (V \cup \{v_0\}, E)$ 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 $\mathcal{Z}(i)$ the set of requests admitted by* `Algorithm` *2 prior to the arrival of request r_i. Then, the sum of usage cost of all cloudlets is,*

$$\sum_{v \in V} w_v(i) \le 2 \cdot |V| \cdot \log_2 \alpha \cdot \sum_{r_{i'} \in \mathcal{Z}(i)} (c(s_{i'}) \cdot u_{i'}), \tag{19}$$

where α is a constant with $2|V| \cdot u_{max} + 2 \le \alpha \le 2^{\frac{C_{min}}{c_{max}}}$, $u_{max} = \lambda - 1$, $C_{min} = min\{C_v \mid v \in V\}$ and $c_{max} = \max\{c(s_i) \mid r_i \in R\}$.

Proof If request $r_{i'} \in R$ is admitted and allocated to cloudlet v' by `Algorithm` 2, we have

$$w_{v'}(i'+1) - w_{v'}(i') = C_{v'} \cdot (\alpha^{1-\frac{C_{v'}(i'+1)}{C_{v'}}} - 1) - C_{v'} \cdot (\alpha^{1-\frac{C_{v'}(i')}{C_{v'}}} - 1)$$
$$= C_{v'} \cdot \alpha^{1-\frac{C_{v'}(i')}{C_{v'}}} \cdot (\alpha^{\frac{C_{v'}(i') - C_{v'}(i'+1)}{C_{v'}}} - 1)$$

Algorithm 2 Online algorithm for the online utility maximization problem

```
Require: An MEC network $G = (V \cup \{v_0\}, E)$ with a set $V$ of APs, each $v \in V$ is attached a cloudlet with computing capacity $C_v$, a set $R$ of user requests $r_i = \langle s_i, l_i, D_i, \beta_i \rangle$ arrived one by one, there is no knowledge of future request arrivals.
Ensure: Maximize the average sum of accumulative utilities of all admitted requests during the $T$ time slots.
1: $A \leftarrow \emptyset$; /* the solution */
2: while request $r_i$ arrives do
3:     for each node $v_j \in V \cup \{v_0\}$ do
4:         Calculate the length $l^t(v_{l_i}, v_j)$ of the shortest path in $G$ between any two nodes $v_{l_i}$ and $v_j$, by the defined link metric (distance or congestion);
5:     end for;
6:     $Q_i \leftarrow \emptyset$; /* the set of candidate cloudlets for $r_i$ */
7:     for each cloudlet $v_j \in V$ do
8:         if cloudlet $v_j$ has sufficient computing resource for $r_i$ then
9:             Calculate its utility gain if request $r_i$ is allocated to cloudlet $v_j$;
10:            if its utility gain is positive then
11:                $Q_i \leftarrow Q_i \cup \{v_j\}$;
12:            end if;
13:        end if;
14:    end for;
15:    if $Q_i = \emptyset$ then
16:        if allocating $r_i$ to remote cloud makes positive utility gain then
17:            Admit $r_i$ by allocating $r_i$ to remote cloud;
18:        else
19:            Reject $r_i$;
20:        end if;
21:    else
22:        Identify the cloudlet $v' \in Q_i$ with the minimum normalized cost by Eq. (18). And calculate the utility gain $u_{i'}$ if request $r_{i'}$ is allocated to cloudlet $v'$;
23:        if $\psi_{v'}(i') > |V| \cdot u_{i'}$ then
24:            if allocating $r_i$ to remote cloud makes positive utility gain then
25:                Admit $r_i$ by allocating $r_i$ to remote cloud;
26:            else
27:                Reject $r_i$;
28:            end if;
29:        else
30:            Admit $r_i$ by allocating $r_i$ to cloudlet $v'$;
31:            Update the residual computing resource of cloudlet $v'$;
32:        end if;
33:    end if;
34:    if $r_i$ is admitted then
35:        $A \leftarrow A \cup \{r_i\}$;
36:    end if
37: end while;
38: return feasible solution $A$ to the online average total utility maximization problem.
```

$$\le C_{v'} \cdot \alpha^{1-\frac{C_{v'}(i')}{C_{v'}}} \cdot (\alpha^{\frac{c(s_{i'})}{C_{v'}}} - 1)$$
$$= C_{v'} \cdot \alpha^{1-\frac{C_{v'}(i')}{C_{v'}}} \cdot (2^{\frac{c(s_{i'})}{C_{v'}} \cdot \log_2 \alpha} - 1)$$
$$\le C_{v'} \cdot \alpha^{1-\frac{C_{v'}(i')}{C_{v'}}} \cdot \frac{c(s_{i'})}{C_{v'}} \cdot \log_2 \alpha. \tag{20}$$
$$= c(s_{i'}) \cdot \alpha^{1-\frac{C_{v'}(i')}{C_{v'}}} \cdot \log_2 \alpha, \tag{21}$$

where Ineq. (20) holds because $2^x - 1 \le x$ with $0 \le x < 1$.

If $r_{i'}$ is not allocated to cloudlet v, the usage cost of cloudlet v does not change. Then the difference in the sums of the usage costs of all cloudlets before and after admitting request $r_{i'}$ thus is

$$\sum_{v \in V} (w_v(i'+1) - w_v(i')) = w_{v'}(i'+1) - w_{v'}(i')$$
$$\le c(s_{i'}) \cdot \alpha^{1-\frac{C_{v'}(i')}{C_{v'}}} \cdot \log_2 \alpha, \text{ by (21)}$$
$$= \log_2 \alpha \cdot c(s_{i'}) \cdot ((\alpha^{1-\frac{C_{v'}(i')}{C_{v'}}} - 1) + 1)$$
$$= \log_2 \alpha \cdot c(s_{i'}) \cdot (\psi_{v'}(i') + 1), \text{ by (18)}$$
$$\le \log_2 \alpha \cdot c(s_{i'}) \cdot (|V| \cdot u_{i'} + 1) \tag{22}$$
$$\le 2 \cdot \log_2 \alpha \cdot |V| \cdot c(s_{i'}) \cdot u_{i'} \tag{23}$$

where Ineq. (22) holds because request $r_{i'}$ is admitted by the admission control policy.

The sum of usage costs of all cloudlets prior to the arrival of request r_i thus is

$$\sum_{v \in V} w_v(i) = \sum_{i'=1}^{i-1} \sum_{v \in V} (w_v(i'+1) - w_v(i'))$$
$$= \sum_{r_{i'} \in \mathcal{Z}(i)} \sum_{v \in V} (w_v(i'+1) - w_v(i'))$$
$$\le \sum_{r_{i'} \in \mathcal{Z}(i)} (2 \cdot \log_2 \alpha \cdot |V| \cdot c(s_{i'}) \cdot u_{i'}), \text{ by (23)}$$
$$= 2 \cdot |V| \cdot \log_2 \alpha \cdot \sum_{r_{i'} \in \mathcal{Z}(i)} (c(s_{i'}) \cdot u_{i'})$$

■

Denote by $\mathcal{D}(i)$ the subset of requests admitted by the optimal solution but rejected by `Algorithm` 2 prior to the arrival of request r_i. Denote by $\mathcal{H}(i)$ the set of requests admitted by both the optimal solution and `Algorithm` 2 prior to the arrival of request r_i.

It can be seen that $\mathcal{D}(i) \cup \mathcal{H}(i)$ is the set of admitted requests by the optimal solution. Then, for each request $r_i \in \mathcal{H}(i)$ we have

$$u_i^* \le \frac{u_{max}}{u_{min}} \cdot u_i, \tag{24}$$

where u_i^* and u_i are the utilities obtained of admitting request r_i by the optimal solution and `Algorithm` 2, respectively, while u_{max} and u_{min} are the maximum and minimum possible utilities for any request, which are constants.

Lemma 6.2. *Given an MEC network $G = (V \cup \{v_0\}, E)$ 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 $\mathcal{D}(i)$ the set of requests admitted by the optimal solution but rejected by* `Algorithm` *2 prior to the arrival of request r_i. Denote by $v_{i'}^*$ the node in the optimal solution to which request $r_{i'} \in \mathcal{D}(i)$ is allocated. We have $v_{i'}^* \in V$, $\forall r_{i'} \in \mathcal{D}(i)$, i.e., the requests in the set $\mathcal{D}(i)$ are allocated to cloudlets instead of the remote cloud in the optimal solution.*

Proof We prove this lemma by contradiction. If it exists a request $r_{i'} \in \mathcal{D}(i)$ being allocated to the remote cloud in the optimal solution. It can be seen that allocating request $r_{i'}$ to the remote cloud makes positive utility gain. Then, request $r_{i'}$ can be at least allocated to the remote cloud in Algorithm 2. However, request $r_{i'}$ is rejected by Algorithm 2 which makes contradiction. ■

Lemma 6.3. *Given an MEC network $G = (V \cup \{v_0\}, E)$ 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 $\mathcal{D}(i)$ the set of requests admitted by the optimal solution but rejected by* `Algorithm` *2 prior to the arrival of request r_i. Denote by $v_{i'}^*$ the node in the optimal solution to which request $r_{i'} \in \mathcal{D}(i)$ is allocated, and denote by $u_{i'}^*$ the utility for request $r_{i'} \in \mathcal{D}(i)$ in the optimal solution. Then, for each request $r_{i'} \in \mathcal{D}(i)$, we have*

$$\psi_{v_{i'}^*}(i') > |V| \cdot \frac{u_{min}}{u_{max}} \cdot u_{i'}^*, \tag{25}$$

when $2|V| \cdot u_{max} + 2 \le \alpha \le 2^{\frac{C_{min}}{c_{max}}}$.

Proof By Lemma 6.2, node $v^*_{i'}$ is a cloudlet. And we show the claim by distinguishing into two cases when request $r_{i'}$ is rejected: Case (1). There is no sufficient computing resource in cloudlet $v^*_{i'}$ to admit request $r_{i'}$ by `Algorithm` 2. Case (2). There is sufficient computing resource in cloudlet $v^*_{i'}$ to admit request $r_{i'}$ in `Algorithm` 2. But the selected cloudlet $v_{i'}$ to admit request r_i violates the admission control policy in `Algorithm` 2 (node $v_{i'}$ could be node $v^*_{i'}$).

Case (1). Because node $v^*_{i'}$ is the cloudlet with no enough computing resource to accommodate request $r_{i'}$ prior to the arrival of request $r_{i'}$ in `Algorithm` 2, i.e., $C_{v^*_{i'}}(i') < c(s_{i'})$. Then we have

$$\begin{aligned}
\psi_{v^*_{i'}}(i') &= \alpha^{1-\frac{C_{v^*_{i'}}(i')}{C_{v^*_{i'}}}} - 1 > \alpha^{1-\frac{c(s_{i'})}{C_{v^*_{i'}}}} - 1 \\
&\ge \alpha^{1-\frac{1}{\log_2 \alpha}} - 1, \text{ since } \alpha \le 2^{\frac{C_{min}}{c_{max}}} \le 2^{\frac{C_{v^*_{i'}}}{c(s_{i'})}} \\
&= \frac{\alpha}{2} - 1 \ge |V| \cdot u^*_{i'}, \text{ since } \alpha \ge 2|V| \cdot u_{max} + 2. \\
&\ge |V| \cdot \frac{u_{min}}{u_{max}} \cdot u^*_{i'}. \qquad (26)
\end{aligned}$$

Case (2).Because we assign request r_i to the cloudlet $v_{i'}$ with the minimum normalized cost by Eq. (18) in `Algorithm` 2 (node $v_{i'}$ could be node $v^*_{i'}$). We then have

$$\psi_{v^*_{i'}}(i') \ge \psi_{v_{i'}}(i'). \qquad (27)$$

Since the admission control policy is violated if request $r_{i'}$ is assigned to cloudlet $v_{i'}$ in Algorithm 2, according to the condition (i) of the admission control policy, we have

$$\psi_{v_{i'}}(i') > |V| \cdot u_{i'} \ge |V| \cdot \frac{u_{min}}{u_{max}} \cdot u^*_{i'}. \qquad (28)$$

Thus, we have

$$\psi_{v^*_{i'}}(i') > |V| \cdot \frac{u_{min}}{u_{max}} \cdot u^*_{i'}. \qquad (29)$$

■

Denote by $\mathbb{P}_{opt}(i)$ and $\mathbb{P}(i)$ the total utility of admitted requests by an optimal solution and `Algorithm` 2 prior to the arrival of request r_i, respectively.

Lemma 6.4. *Given an MEC network* $G = (V \cup \{v_0\}, E)$ *and a finite time horizon that consists of* $T \ge 1$ *time slots, let* R *be the set of requests arriving one by one over the time horizon, denote by* $\mathcal{D}(i)$ *the set of requests admitted by the optimal solution but rejected by* `Algorithm` *2 prior to the arrival of request* r_i*. We have*

$$\mathbb{P}_{opt}(i) \le \frac{u_{max}}{u_{min}} \cdot \mathbb{P}(i) + \sum_{r_{i'} \in \mathcal{D}(i)} u^*_{i'}. \qquad (30)$$

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

$$\begin{aligned}
\mathbb{P}_{opt}(i) &= \sum_{r_{i'} \in \mathcal{H}(i)} u^*_{i'} + \sum_{r_{i'} \in \mathcal{D}(i)} u^*_{i'} \\
&\le \frac{u_{max}}{u_{min}} \cdot \sum_{r_{i'} \in \mathcal{H}(i)} u_{i'} + \sum_{r_{i'} \in \mathcal{D}(i)} u^*_{i'}, \text{ by (24)} \\
&\le \frac{u_{max}}{u_{min}} \cdot \mathbb{P}(i) + \sum_{r_{i'} \in \mathcal{D}(i)} u^*_{i'}. \qquad (31)
\end{aligned}$$

Ineq. (31) holds because $\mathcal{H}(i)$ is the subset of admitted requests by `Algorithm` 2. ■

Theorem 6.5. *Given an MEC network* $G = (V \cup \{v_0\}, E)$ *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, there is an online algorithm for the online average total utility maximization problem,* `Algorithm` *2, with a competitive ratio of* $O(\log|V|)$*, which takes* $O(|V|^2)$ *time to admit each request when* $\alpha = 2|V| \cdot u_{max} + 2$*, where* V *is the set of cloudlet nodes.*

Proof

$$\begin{aligned}
&|V| \cdot (\mathbb{P}_{opt}(i) - \frac{u_{max}}{u_{min}} \cdot \mathbb{P}(i)) \le |V| \cdot \sum_{r_{i'} \in \mathcal{D}(i)} u^*_{i'}, \text{ by Lemma 6.4,} \\
&= \sum_{r_{i'} \in \mathcal{D}(i)} |V| \cdot u^*_{i'} < \frac{u_{max}}{u_{min}} \cdot \sum_{r_{i'} \in \mathcal{D}(i)} \psi_{v^*_{i'}}(i'), \text{ by Lemma 6.3} \\
&\le \frac{u_{max}}{u_{min}} \cdot \sum_{r_{i'} \in \mathcal{D}(i)} \psi_{v^*_{i'}}(i) \qquad (32) \\
&= \frac{u_{max}}{u_{min}} \cdot \sum_{r_{i'} \in \mathcal{D}(i)} \frac{w_{v^*_{i'}}(i)}{C_{v^*_{i'}}} \le \frac{u_{max}}{u_{min}} \cdot \sum_{r_{i'} \in \mathcal{D}(i)} \sum_{v \in V} \frac{w_v(i)}{C_v} \\
&\le \frac{u_{max}}{u_{min}} \cdot \sum_{v \in V} w_v(i) \sum_{r_{i'} \in \mathcal{D}(i)} \frac{1}{C_v} \qquad (33) \\
&\le \frac{u_{max}}{u_{min}} \cdot \sum_{v \in V} w_v(i) \cdot 1 = \frac{u_{max}}{u_{min}} \cdot \sum_{v \in V} w_v(i) \\
&\le 2 \cdot |V| \cdot \frac{u_{max}}{u_{min}} \cdot \log_2 \alpha \cdot \sum_{r_{i'} \in \mathcal{Z}(i)} (c(s_{i'}) \cdot u_{i'}), \text{ by Lemma 6.1,}
\end{aligned}$$

where Ineq. (32) holds, because the utilization of the computing resource does not decrease. And Ineq. (33) holds, because $\sum_{i=1}^{p} \sum_{j=1}^{q} A_i B_j \le \sum_{i=1}^{p} A_i \sum_{j=1}^{q} B_j$.

We thus have

$$\begin{aligned}
\mathbb{P}_{opt}(i) &< \frac{u_{max}}{u_{min}} \cdot \mathbb{P}(i) + 2 \cdot \frac{u_{max}}{u_{min}} \cdot \log_2 \alpha \cdot \sum_{r_{i'} \in \mathcal{Z}(i)} (c(s_{i'}) \cdot u_{i'}) \\
&\le \frac{u_{max}}{u_{min}} \cdot \mathbb{P}(i) + 2 \cdot \frac{u_{max}}{u_{min}} \cdot \log_2 \alpha \cdot c_{max} \cdot \sum_{r_{i'} \in \mathcal{Z}(i)} \cdot u_{i'}.
\end{aligned}$$

Then,

$$\begin{aligned}
&\frac{\mathbb{P}_{opt}(i)}{\mathbb{P}(i)} < \frac{\frac{u_{max}}{u_{min}} \cdot \mathbb{P}(i) + 2 \cdot \frac{u_{max}}{u_{min}} \cdot \log_2 \alpha \cdot c_{max} \cdot \sum_{r_{i'} \in \mathcal{Z}(i)} u_{i'}}{\mathbb{P}(i)} \\
&= \frac{\frac{u_{max}}{u_{min}} \cdot \sum_{r_{i'} \in \mathcal{Z}(i)} u_{i'} + 2 \cdot \frac{u_{max}}{u_{min}} \cdot \log_2 \alpha \cdot c_{max} \cdot \sum_{r_{i'} \in \mathcal{Z}(i)} u_{i'}}{\sum_{r_{i'} \in \mathcal{Z}(i)} u_{i'}} \\
&= \frac{u_{max}}{u_{min}} + 2 \cdot \frac{u_{max}}{u_{min}} \cdot \log_2 \alpha \cdot c_{max} \\
&= O(\log|V|), \text{ when } \alpha = 2|V| \cdot u_{max} + 2.
\end{aligned}$$

The time complexity of `Algorithm` 2 is dominated by finding the shortest paths (delay) from node v_{l_i} to other nodes, which takes $O(|V|^2)$ time. ■

7 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.

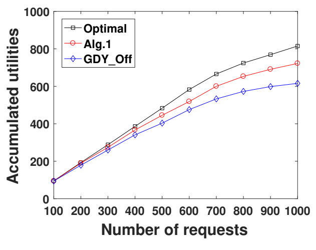
7.1 Environment setting

We consider a heterogeneous MEC network consisting of 200 APs, and 10% of which are co-located with cloudlets. The topologies of MEC networks are generated by GT-ITM [3]. For each AP, the bandwidth at each time slot is drawn from 20 MHz to 40 MHz randomly [9], the signal-to-noise ratio (i.e., $\frac{P_i}{R^{\alpha}}$) of an AP is set as 30 dB [11]. For each cloudlet, the capacity varies from $3,000$ MHz to $7,000$ MHz [14] and its processing rate varies from 0.5 MB to 2 MB per millisecond [7]. For each request, its task size is randomly drawn from 1 MB to 5 MB [8], the demanded computing resource is randomly drawn from 20 MHz to 300 MHz [13] and the delay requirement is randomly drawn from 10 ms to 50 ms [7]. The transmission delay of a link at each time slot is chosen from 2 ms to 5 ms randomly [14], while the transmission delay from an AP to the remote cloud varies from 80 ms to 100 ms. We further assume the processing rate of the remote cloud is 20 MB per ms. λ is set as 2 and β_i ranges from 1 to 3. We assume 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 [7]. The result in each figure is the mean of the results by applying an algorithm on 20 MEC network instances of the same size, and the running time of each algorithm is obtained, based on a desktop with a 3.60 GHz Intel 8-Core i7-7700 CPU and 16 GB RAM. Unless specified, the above parameters are adopted by default.

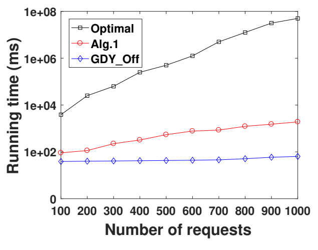
We first investigated the proposed algorithm `Algorithm 1` (referred to as `Alg.1`) under the offline version over the first time slot against two benchmarks. One is its ILP solution (7) obtained by exhaust search (referred to as `Optimal`). The other is a greedy algorithm (referred to as `GRD_Off`), where we assign each request to the cloudlet (or the remote cloud) with the largest utility one by one.

We then evaluated the proposed algorithm `Algorithm 2` (referred to as `Alg.2`) under the online version over the time horizon (i.e., 100 time slots) against a greedy algorithm (referred to as `GRD_On`), which is the online version of `GRD_Off`.

7.2 Performance evaluation of the proposed algorithms



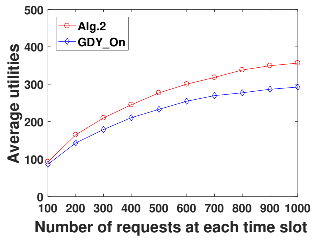
(a) The accumulated utility



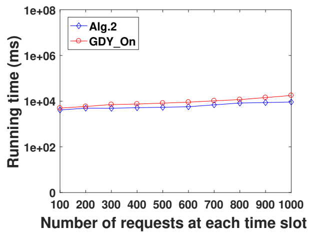
(b) The running time

Figure 1: Performance of different algorithms for the total utility maximization problem.

We first studied the performance of `Alg.1` against algorithms `Optimal` and `GRD_Off`, by varying the number of requests from 100 to $1,000$. We then evaluated the performance of `Alg.2` against algorithm `GRD_On`, by varying the number of requests from 100 to $1,000$ arriving at each time slot. Fig. 1 depicts the accumulated utilities and running time of different algorithms for the total utility maximization problem, and Fig. 2 depicts the average utilities and the running time of different algorithms for the online average total utility maximization problem. It can be seen from Fig. 1(a) that when the number of requests reaches $1,000$, the performance achieved by algorithm `GRD_Off` is 88.5% of that by `Alg.1` while the performance achieved by `Alg.1` is 85.2% of that by algorithm `Optimal`. The similar performance behaviors can be found in Fig. 2(a). The rationale behind is that both `Alg.1` and `Alg.2` better utilize the network resource by provisioning satisfied services to more users, compared with the greedy algorithms.



(a) The average utility



(b) The running time

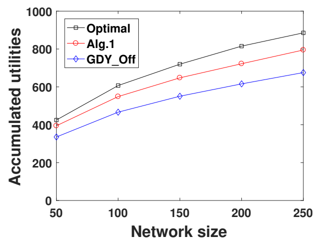
Figure 2: Performance of different algorithms for the online average total utility maximization problem.

7.3 Impact of important parameters on the performance of the proposed algorithms

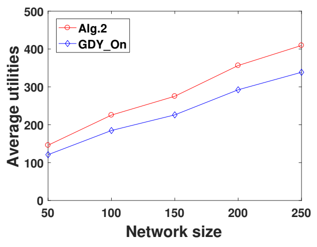
The rest is to investigate the impact of important parameters on the performance of the proposed algorithms including network size, parameter β_i, and parameter α. We also study the performance of the online algorithm with and without adopting the admission control policy.

We first studied the impact of network size on the proposed algorithms, by varying the number of APs from 50 to 250. Recall that 10% of APs are co-located with cloudlets. Fig. 3(a) depicts the accumulated utilities by different algorithms for the total utility maximization problem. And Fig. 3(b) depicts the average utilities of different algorithms for the online average total utility maximization problem. We can see from Fig. 3(a) that when the network size is 250, the performance achieved by algorithm `GRD_Off` is 76.3% of that by `Alg.1` while the performance achieved by `Alg.1` is 84.8% of that by `Optimal`. The similar performance behaviors can be observed in Fig. 3(b) 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.

We then evaluated the impact of parameter β_i on the performance of the proposed algorithms, by varying the network size from 50 to 250. Fig. 4 illustrates the impact of parameter β_i on the proposed algorithms `Alg.1` and `Alg.2` when $\beta_i = 1$, 2, and 3 respectively. It can be seen from Fig. 4(a) that when the network size is 250, the performance of `Alg.1` with $\beta_i = 1$ is 54.9% of itself with $\beta_i = 3$. And when the network size is 250, the performance of `Alg.1` with $\beta_i = 1$ is 86.7% of itself with $\beta_i = 3$. The rationale behind is that a larger β_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



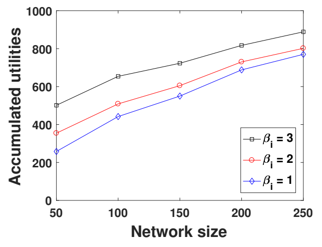
(a) Different algorithms for the total utility maximization problem



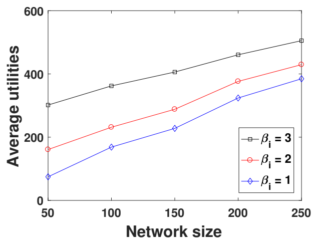
(b) Different algorithms for the online average total utility maximization problem

Figure 3: The impact of network size on the performance of the proposed algorithms

their requests that result in longer service delays. Thus, a larger β_i is important in admitting requests when the network size is small. The similar performance behavior can be found in Fig. 4(b).



(a) `Alg.1` for the total utility maximization problem



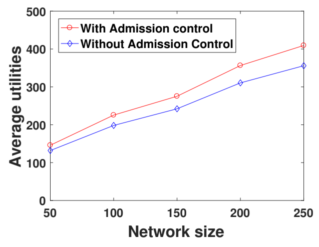
(b) `Alg.2` for the online average total utility maximization problem

Figure 4: The impact of β_i on the performance of the proposed algorithms.

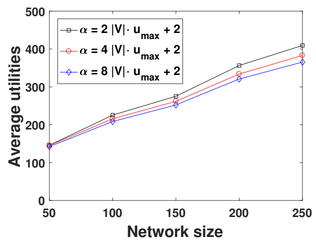
We finally investigated both the impact of the admission control policy on the performance of `Alg.2`, by varying the value of parameter α and the network size from 50 to 250. Fig. 5(a) depicts the performance of `Alg.2` with and without the admission control policy. It can be seen from Fig. 5(a) that when the network size is 250, the performance of `Alg.2` without the admission control policy is 86.9% of itself with the admission control policy. 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. Fig.5(b) demonstrates the performance of `Alg.2` with parameter $\alpha = 2|V| \cdot u_{max} + 2$, $4|V| \cdot u_{max} + 2$, and $8|V| \cdot u_{max} + 2$, respectively, where $|V|$ is the network size, and $u_{max} = \lambda - 1$ is the maximum possible utility gain for a request. It can be from Fig.5(b) that when the network size is 250, the performance of `Alg.2` with $\alpha = 8|V| \cdot u_{max} + 2$ is 89.3% of itself with $\alpha = 2|V| \cdot u_{max} + 2$. The justification is that with a larger α, the normalized cost of computing resource becomes higher by Eq. (18), and it intends to be conservative and reject requests.

8 CONCLUSIONS

In this paper, we studied the delay-sensitive service provisioning problem for IoT applications in an edge computing environment, by offloading task service requests to either the remote cloud or local cloudlets in an MEC network. We first formulated two novel optimization problems and showed the NP-hardness of the defined problems. We then proposed an approximation algorithm with a provable approximation ratio for the total utility maximization problem. We also developed an online algorithm with a provable competitive ratio for dynamic admissions of task offloading requests without the knowledge of future request arrivals. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.



(a) With and without the admission control policy



(b) Different values of α

Figure 5: The impacts of the admission control policy and parameter α on the performance of algorithm `Alg.2`.

REFERENCES

[1] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. Mobile edge computing: a survey. *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450 – 465, 2018.
[2] Reuven Cohen, Liran Katzir, and Danny Raz. An efficient approximation for the generalized assignment problem. *Information Processing Letters*, Vol. 100, pp. 162–166, 2006.
[3] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/, 2019.
[4] Mike Jia, Jiannong Cao, and Weifa Liang,. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, vol.5, no. 4, pp.725–737, 2017.
[5] Mike Jia, Weifa Liang, Zichuan Xu, and Meitian Huang. Cloudlet load balancing in wireless metropolitan area networks. *Proc of INFOCOM'16*, pp.730–738, IEEE, 2016.
[6] Mike Jia, Weifa Liang, Zichuan Xu, Meitian Huang, and Yu Ma. QoS-aware cloudlet load balancing in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, vol.8, no.2, pp. 623–634, 2020.
[7] Yu Ma, Weifa Liang, Zichuan Xu, and Song Guo. Profit maximization for admitting requests with network function services in distributed clouds. *IEEE Transactions on Parallel and Distributed Systems*, vol.30, no. 5, pp. 1143 – 1157, 2019.
[8] Yaozhong Song, Stephen S. Yau, Ruozhou Yu, Xiang Zhang, and Guoliang Xue. An approach to QoS-based task distribution in edge computing networks for IoT applications. *Proc of the International Conference on Edge Computing*(EDGE), pp.32 –39, IEEE, 2017.
[9] Tuyen X. Tran, and Dario Pompili. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2018.
[10] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication.* Cambridge University Press, 2005.
[11] Wanqing Wu, Sandeep Pirbhulal, Arun Kumar Sangaiah, Subhas Chandra Mukhopadhyay, and Guanglin Li. Optimization of signal quality over comfortability of textile electrodes for ECG monitoring in fog computing based medical applications. *Future generation computer systems*, vol. 86, pp. 515 – 526, 2018.
[12] Qiufen Xia, Weifa Liang, and Wenzheng Xu. Throughput maximization for online request admissions in mobile cloudlets. *Proc of 38th Annual IEEE Conference on Local Computer Networks* (LCN), IEEE, 2013.
[13] Zichuan Xu, Weifa Liang, Wenzheng Xu, Mike Jia, and Song Guo. Efficient algorithms for capacitated cloudlet placements. *IEEE Transactions on Parallel and Distributed Systems*, vol.27, no.10, pp.2866 – 2880, 2016.
[14] Zichuan Xu, Weifa Liang, Mike Jia, Meitian Huang, and Guoqiang Mao. Task offloading with network function services in a mobile edge-cloud network. *IEEE Transactions on Mobile Computing*, vol.18, no. 11, pp.2672–2685, 2019.
[15] Zichuan Xu, Wanli Gong, Qiufen Xia, Weifa Liang, Omer F. Rana, and Guowei Wu. NFV-enabled IoT service provisioning in mobile edge clouds. *IEEE Transactions on Mobile Computing*, to be published, 2020, doi: 10.1109/TMC.2020.2972530
[16] Zichuan Xu, Zhiheng Zhang, Weifa Liang, Qiufen Xia, Omer F. Rana, and Guowei Wu. QoS-aware VNF placement and service chaining for IoT applications in multi-tier mobile edge networks. *ACM Transactions on Sensor Networks*, vol.16, no.3, Article 23:1–23:27, 2020.
[17] Ruozhou Yu, Guoliang Xue, and Xiang Zhang. Application provisioning in fog computing-enabled internet-of-things: a network perspective. *Proc of INFOCOM'18*, pp.783 –791, IEEE, 2018.