Efficient Algorithms for Capacitated Cloudlet Placements

Zichuan Xu, *Student Member, IEEE*, Weifa Liang, *Senior Member, IEEE*, Wenzheng Xu, *Member, IEEE*, Mike Jia, and Song Guo, *Senior Member, IEEE*

Abstract—Mobile cloud computing is emerging as a main ubiquitous computing platform to provide rich cloud resources for various applications of mobile devices. Although most existing studies in mobile cloud computing focus on energy savings of mobile devices by offloading computing-intensive jobs from mobile devices to remote clouds, the access delays between mobile users and remote clouds usually are long and sometimes unbearable. Cloudlet as a new technology is capable to bridge this gap, and can enhance the performance of mobile devices significantly while meeting the crisp response time requirements of mobile users. In this paper, we study the cloudlet placement problem in a large-scale Wireless Metropolitan Area Network (WMAN) consisting of many wireless Access Points (APs). We first formulate the problem as a novel capacitated cloudlet placement problem that places K cloudlets to some strategic locations in the WMAN with the objective to minimize the average access delay between mobile users and the cloudlets serving the users. We then propose an exact solution to the problem by formulating it as an Integer Linear Programming (ILP). Due to the poor scalability of the ILP, we instead propose an efficient heuristic for the problem. For a special case of the problem where all cloudlets have identical computing capacities, we devise novel approximation algorithms with guaranteed approximation ratios. We also devise an online algorithm for dynamically allocating user requests to different cloudlets, if the K cloudlets have already been placed. We finally evaluate the performance of the proposed algorithms through experimental simulations. Simulation results demonstrate that the proposed algorithms are promising and scalable.

Index Terms—Cloudlet placement, cloudlet access delay minimization, mobile user request assignment, mobile cloud computing, approximation algorithms

1 Introduction

In recent years, mobile devices have undergone a transformation from bulky gadgets with limited functionalities to indispensable everyday accessories. Advances in mobile hardware technology have led to an explosive growth in mobile application markets. Although mobile applications are becoming increasingly computational-intensive, the computing capacity of mobile devices remains limited, due to the considerations of weight, size, battery life, ergonomics, and heat dissipation of portable mobile devices [30]. A powerful approach to enhancing the performance of mobile applications is enabling mobile devices to offload some of the workload of mobile devices to remote resource-rich clouds [18], [39]. Although clouds have rich computing and storage resources, they are geographically far away from mobile users. Communication delays between the clouds and their mobile users thus can be long and unpredictable. This is especially problematic for mobile applications in which a crisp response time is critical to their users, such as augmented reality, speech recognition, navigation, language translation, etc. [30], [32]. To reduce this long access delay, cloudlets were proposed as an alternative solution [30] to powerful remote clouds. Cloudlets are resource-rich server clusters co-located with wireless Access Points (APs) in a local network, and mobile users can offload their tasks to local cloudlets for processing [8], [30], [33]. As cloudlets are self-managing, with fewer requirements other than power and Internet connectivity, they can be deployed in existing networks, leading them to be viewed as the 'data centers in a box'. The physical proximity between mobile users and cloudlets means that the cloudlet access delay on task offloading can be greatly reduced, compared to remote clouds, thereby significantly improving mobile user experiences.

Most existing studies focused on offloading tasks of mobile users to cloudlets for energy savings of mobile devices, assuming that the cloudlets have already been placed [7], [10], [18], [19], [34], [35], [39]. Little attention has been paid to cloudlet placements and the impact of different placements on mobile users. In contrast, we focus on the cloudlet placements in a Wireless Metropolitan Area Network (WMAN) that provides wireless Internet coverage for mobile users in a large-scale metropolitan area, where the WMAN is often owned and operated by local governments as public infrastructures [21]. This will bring the following benefits: (1) the metropolitan area covered by the WMAN has a high population density, meaning that the cloudlets will be accessible by a large number of mobile users; (2) due to the network size of the WMAN, service providers can take advantage of the economics of scale when offering cloudlet services through the WMAN, making cloudlet services more affordable to the

- *Z. Xu, W. Liang, and M. Jia are with the Research School of Computer Science, the Australian National University, Canberra, ACT 0200, Australia. E-mail: {edward.xu, u5515287}@anu.edu.au, wliang@cs.anu.edu.au.*
- *W. Xu is with the College of Computer Science, Sichuan University, Chengdu 610065, P.R. China. E-mail: wenzheng.xu3@gmail.com.*
- *S. Guo is with the School of Computer Science and Engineering, University of Aizu, Aizu-Wakamatsu City, Fukushima 965-8580, Japan. E-mail: sguo@u-aizu.ac.jp.*

Manuscript received 14 Aug. 2015; revised 25 Nov. 2015; accepted 1 Dec. 2015. Date of publication 22 Dec. 2015; date of current version 14 Sept. 2016. Recommended for acceptance by H. Jin.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2015.2510638

general public. However, placing cloudlets in a WMAN is challenging. The locations of cloudlets are critical to the access delays of mobile users and the resource utilization of cloudlets, especially in a large-scale WMAN that consists of hundreds and thousands of Access Points, where mobile users access the cloudlets through their local APs. Due to the large size of the WMAN, poor cloudlet placements will result in long access delays and heavily unbalanced load among cloudlets, i.e., some of the cloudlets are overloaded while others are under-loaded and even idle. Therefore, strategically placing capacitated cloudlets will significantly improve the performance of various mobile applications such as the average cloudlet access delay. We will focus on a novel cloudlet placement problem in a large-scale WMAN, where a cloudlet service provider is planning to deploy K (≥ 1) cloudlets at some strategic AP locations in a WMAN for mobile user access. The objective is to minimize the average cloudlet access delay between the mobile users and the cloudlets serving the users. The challenge associated with such placements are that: which cloudlets should be placed to which locations, and which user requests should be assigned to which cloudlets so that the average cloudlet access delay among the mobile users is minimized. As the problem is NP-hard, is there any approximation algorithm with a guaranteed approximation ratio for it? In this paper we will address these issues.

There are several placement problems in networks such as cache placements and server placements that have been studied in the past decades [26], [38]. For example, the cache placement problem is to choose K replicas or hosting services among N potential sites, such that the latency experienced by users is minimized [38], which usually is reduced to the capacitated K-median problem. Due to the NP-hardness of the latter, there are approximation algorithms for unsplittable and splittable versions of the problem [6], [23], where 'unsplittable' refers to that a user request can be served by only one data center [6], and 'splittable' indicates that the user request can be served by multiple centers [23]. In spite of some similarities between the cache/server placement problem [26], [38] and the cloudlet placement problem, they are essentially different. First, existing studies assumed that either there is no capacity constraint on caches/servers or the capacity of each cache/server is identical. In contrast, we here consider cloudlets with different computing capacity constraints, as cloudlets are differently configured to meet the demands of different applications in the real world [15], [35]. Second, existing studies simply assumed that all user requests have identical amounts of resource demands, while we assume that different user requests may have different amounts of resource demands. Thus, the existing solutions to cache/server placement problem [6] may not be applicable to the cloudlet placement problem. Particularly, the number of user requests in a large-scale WMAN usually is several orders of magnitudes of the network size, e.g., there are several million mobile users in a metropolitan area, compared with only several hundred nodes (access points) in such a network. Therefore, new algorithms for cloudlet placement problem must be devised in order to deal with a large number of user requests and different amounts of resource demands by different users.

The main contributions of this paper are as follows. We study multiple cloudlet placements in a large-scale WMAN, by formulating a novel capacitated cloudlet placement problem with the objective of minimizing the average cloudlet access delay. We first show that the problem is NP-hard, and propose an exact solution by formulating it as an Integer Linear Programming (ILP). Due to the poor scalability of the ILP, we then devise a fast, scalable heuristic. For a special case of the problem where all cloudlets have identical computing capabilities, we devise two approximation algorithms with guaranteed approximation ratios, depending on whether all user requests have identical resource demands or not. We also propose an efficient online algorithm for dynamic user request assignment to the cloudlets, provided the K cloudlets have already been placed. We finally evaluate the performance of the proposed algorithms through experimental simulations. The simulation results demonstrate that the proposed algorithms are very promising.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 introduces the system model and problem definitions. Section 4 provides an ILP solution and a fast yet scalable heuristic. Section 5 devises two approximation algorithms for the problem when all cloudlets have identical computing capacities, depending on whether each user request has different computing resource demands. Section 6 devises an online algorithm to assign user requests to different cloudlets if all the cloudlets have already been placed. Section 7 evaluates the performance of the proposed algorithms by experimental simulation, and Section 8 concludes the paper.

2 Related Work

Most existing studies focused on offloading user tasks to remote clouds by exploring different pricing models and efficient task scheduling [12], [16], [18], [25], [29], [36], [39]. However, the average access delay between users and remote clouds can be prohibitively long. Instead, cloudlets deployed in the vicinities of users have been quickly gaining recognition as alternative offloading destinations due to the short response time and capability of reducing the energy consumption of mobile devices [7], [10], [14], [15], [19], [20], [27], [28], [31], [34], [35], [37]. Cloudlet based mobile computing now is also referred to as fog computing with the aim to reduce the access latency between mobile users and remote clouds, by providing compute, storage, and networking services within the proximity of mobile users [2], [22]. For example, the system Odessa [27] was designed to enable interactive applications on mobile devices while satisfying crisp response time requirements of applications. Chun et al. [7] proposed a system CloneCloud that aims to partition applications between mobile devices and a local cloudlet. Hoang et al. [14] proposed a linear programming (LP) solution for task offloading by considering the QoS requirements of mobile users with an aim to maximize the revenue of service providers. Xia et al. [34], [35] devised novel online algorithms for dynamically admitting user requests to a cloudlet. Cardellini et al. [5] considered that mobile users can offload their tasks to both remote clouds and local cloudlets at the same time, and proposed a game theory-based solution. Similarly, Gelenbe et al. [11] dealt with offloading jobs between a local cloud and a remote cloud, by incorporating energy consumption and quality of service (QoS) criteria. Furthermore,

cloudlets have important impacts on the mobile cloud gaming industry [3], [4]. A cloudlet-assisted multi-player cloud gaming system [4] was reported, by making use of cloudlets as caches for video frames. Despite the increasing momentum of cloudlet research, the question of where cloudlets should be placed within a network has largely been overlooked. Previous works typically assumed that cloudlets are used in small private WLANs such as in campuses, buildings, or even at office floors. In such a setting, it can be argued that the placement of cloudlets is trivial. Wherever the cloudlet is placed, the small network size implies that the communication delays between the cloudlets and its users are negligible. However, the cloudlet placement in a large-scale WMAN is non-trivial, as there are millions of users with hundreds and thousands of APs in the network. In such a large-scale WMAN, the average cloudlet access delay between a user and the cloudlet serving the user could be prohibitively long, and the user may not be tolerable to such a long delay. As a result, to minimize the average cloudlet access delay, developing efficient algorithms for multiple cloudlet placements in the WMAN becomes imperative.

There are two classical optimization problems closely related to the cloudlet placement problem: the *cache placement problem* [38] and the *server placement problem* [26]. The former is to choose K replicas or hosting services among N potential sites to minimize the latency experienced by users [38], and the latter is to place a number of server replicas among some potential locations such that the perceived delay of users is minimized under a given traffic pattern. Both of these two problems can be solved by a direct reduction to the capacitated K-median problem [6]. Intuitively, the cloudlet placement problem can also be reduced to the capacitated K-median problem. However, the problem of concern is essentially different from the two mentioned problems: one is that there is either no capacity constraints on cache and servers, or all caches and servers have identical capacities, while we here assume that different cloudlets have different capacities and different user requests may have different computing resource demands. Another is that directly adopting the approximation algorithm in [6] for the cloudlet placement problem in on a large-scale WMAN will result in poor scalability, by treating each user request at each AP as a virtual user request, since the number of user requests in the network can be several orders of magnitudes of the network size.

3 Preliminaries

In this section we introduce the system model, define the problems precisely, and show the NP-hardness of the cloudlet placement problem.

3.1 System Model

We consider a WMAN $G = (V \cup S, E)$ consisting of many APs and a set of potential locations for cloudlets, where V is the set of APs and S is the set of potential locations of cloudlets, and E is the set of links between two APs or between an AP and a cloudlet at a location in S. Denote by n and m the numbers of APs and links in V and E, respectively, i.e., $n = |V|$ and $m = |E|$. For each AP v_j in V, let $w(v_j)$ represent *the expected number of user requests* using the AP to access

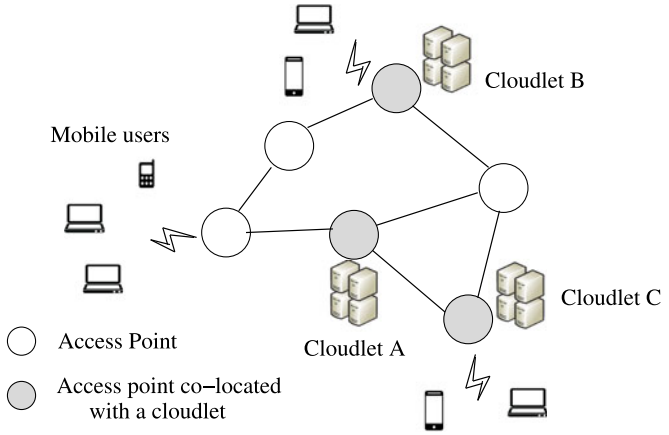


Fig. 1. A WMAN $G = (V, E)$ with $K = 3$ cloudlets.

cloudlets in the network, which is a positive integer. As APs usually are deployed at strategic locations such as shopping malls, train stations, schools, libraries, etc., the number of user requests $w(v)$ at each AP v per unit time can be estimated by the population density in that area, or the historic AP access information through a linear regression technique. Let $\mathcal{R}_j$ be a set of user requests at AP v_j with $w(v_j) = |\mathcal{R}_j|$, and different user requests in $\mathcal{R}_j$ may have different amounts of computing resource demands. Denote by r_m a user request in $\mathcal{R}_j$ with the computing resource demand γ_m in G. In addition, for each link (v_j, v_l) in E, denote by d_{jl} the data transmission latency between its two endpoints (APs) v_j and v_l.

Assume that there are K cloudlets to be placed to K different locations in S. For simplicity, we assume that the cloudlets will be co-located with some APs, i.e., $S \subseteq V$. We further assume that $K \ll |S|$ and each cloudlet C_i has limited computing resource to process user requests. Let c_i be the computing resource capacity of C_i with $1 \leq i \leq K$. Given the K placed cloudlets, mobile users can offload their tasks to the cloudlets through their local APs. If a cloudlet is co-located with an AP, the users at that AP will have the minimum cloudlet access delay of the users; otherwise, the user requests at that AP must be relayed to nearby cloudlets for processing, resulting in a cloudlet access delay due to the accumulative delay of multiple hop relays. Fig. 1 illustrates a WMAN network.

3.2 Problem Definitions

The capacitated cloudlet placement problem in a WMAN $G = (V \cup S, E)$ is defined as follows. Given the network G, a set $\{C_1, C_2, \ldots, C_K\}$ of K (≥ 1) cloudlets with capacities $c_1, c_2, \ldots, c_K$, respectively to be placed to K locations in a set S of potential locations with $S \subseteq V$, a set $\mathcal{R}_j$ of user requests at each AP $v_j \in V$, the number of user requests $w(v_j)$ in $\mathcal{R}_j$ (i.e., $w(v_j) = |\mathcal{R}_j|$), and the computing resource demand γ_m of each user request $r_m \in \mathcal{R}_j$, assume that the delay of data transmission on links in E is defined as $d: E \mapsto \mathbb{R}^{\geq 0}$, the problem is to place the K cloudlets to the K locations in S such that the average cloudlet access delay between the mobile users and the cloudlets serving their requests is minimized, subject to that the accumulative computing resource allocated to all user requests in each cloudlet is no more than its capacity, where the cloudlet access delay d_{jl} between a user request at AP $v_j \in V$ and a cloudlet at

TABLE 1
Symbols

Symbols	Notations
$G = (V \cup S, E)$	a WMAN that consists of a set V of APs and a set S of potential locations for cloudlets
$n\ (= \|V\|)$, $m\ (= \|E\|)$	the numbers of APs in V and links in E
v	an AP in V
$\mathcal{R}_j$ and r_m	a set of user requests at AP v_j and a user request in the set
γ_m	the computing resource demand of user request r_m
γ_{min} and γ_{max}	the minimum and maximum computing resource demands of all requests
ρ and ρ_m	$\rho = \frac{\gamma_{max}}{\gamma_{min}}$ and $\rho_m = \frac{\gamma_m}{\gamma_{min}}$
$w(v)$	an integer weight of AP v that represents the number of user requests at v
K and C_i	the number of cloudlets that need to be placed into G and the cloudlet C_i
$c_1, c_2, \ldots, c_K$	the computing capacities of the K cloudlets
$G' = (V', E')$	a complete metric undirected graph in the metric K-median problem with K centers
$v_j \in V'$	a location in V', and there is a client at location v_j
d_{ij}	the service cost for the center at location v_i serving the client at location v_j
p_{il}	a binary variable, where p_{il} implies that cloudlet C_i is placed at location $v_l \in S$
x_{im}	an indicator variable that show whether request r_m is assigned to cloudlet C_i for processing
z_{ij}	the number of user requests at AP v_j that are sent to cloudlet C_i for processing
N_0	a basic unit (block) of the number of user requests of each AP
$n_j\ (= \lfloor w(v_j)/N_0 \rfloor)$	the number of units that each AP v_j has
$c'\ (= \lceil c/N_0 \rceil)$	the capacity of each cloudlet C_i in terms of units of requests
$G_U = (V_U, E_U)$	an auxiliary complete graph from the original graph $G = (V, E)$
$v_j^1, v_j^2, \ldots, v_j^{n_j}$	a set of n_j virtual nodes for each AP v_j
n_j	the number of split blocks of each AP v_j, and
n_r and B	the total number and the set of blocks of all APs in V
$e = (v_i^x, v_j^y)\ (\in E_U)$	a edge between nodes v_i^x and v_j^y, and its weight is the delay between v_i and v_j for transmitting N_0 user requests
$I = \{i_1, i_2, ..., i_K\}$	the set of locations where K cloudlets are placed
$G_f = (\{s\} \cup B \cup I \cup \{t\}, E_f)$	an auxiliary flow graph build from $G_U = (V_U, E_U)$
f and f^*	a flow from s to t in G_f and an integral minimum cost maximum flow in G_f
$D(f^*)$	the cost of flow f^*
D_j	the total delay incurred by the assignment of $n_j N_0$ user requests at AP v_j
ϵ	$= \max_{v_j \in V}\{\frac{1}{\lfloor w(v_j)/N_0 \rfloor}\}$
δ	$= \frac{N_0}{c}$
τ and τ^*	the average cloudlet access delay by algorithm `Appro-Extension` and the optimal solution
τ' and τ'^*	the average cloudlet access delay by algorithm `Appro` through treating each request r_m as ρ_m virtual user requests and the optimal one

location v_l is the length of the shortest path between them. In other words, the problem is to identify K locations from $|S|$ ($\geq K$) potential locations and place the K cloudlets and to determine at which location each cloudlet C_i with capacity c_i should be placed such that the average cloudlet access delay of user requests is minimized.

Assume that the K cloudlets have already been placed into the K locations of S in $G = (V \cup S, E)$, let $S' = \{v_1, v_2, \ldots, v_K\} \subseteq S$ with cloudlet C_i being placed at location $v_i \in S'$, user requests with different computing resource demands arrive at the system dynamically. Since the number of user requests at each AP v_j may vary over time, the workloads of different cloudlets are different over time, some of them may be overloaded while others are under-loaded or idle. Assuming that time is slotted into equal time slots, the arrived user requests in the system will be assigned to the K placed cloudlets for processing in the beginning of each time slot t. Let $w(v_j, t)$ be the number of arrived user requests at AP v_j at time slot t. *The dynamic user request assignment problem* is to assign user requests from different APs in V to the placed K cloudlets in the beginning of time slot t such that as many user requests as possible are assigned to the cloudlets while the average access delay among all assigned requests is minimized.

Table 1 summarizes the symbols used in this paper.

3.3 NP-Hardness

We show the capacitated cloudlet placement problem is NP-hard by a polynomial reduction from another NP-hard problem—the metric K-median problem. Given a complete metric undirected graph $G' = (V', E')$ with one client at each location $v_j \in V'$, the service cost d_{ij} for the center at location v_i serving the client at location v_j, and K centers, *the metric K-median problem* is to place the K centers into K locations in V' such that the total service cost of serving all clients to their centers is minimized [6].

Lemma 1. *The capacitated cloudlet placement problem in a WMAN $G = (V \cup S, E)$ is NP-hard.*

Proof. We reduce the metric K-median problem to the capacitated cloudlet placement problem as follows. Consider the metric K-median problem in a given metric complete graph $G' = (V', E')$. We construct a WMAN $G = (V \cup S, E)$ from G', where $V = V'$, $S = V'$ and $E = E'$. There is an AP at each location $v_j \in V$ with only one user request there. We now consider placing K cloudlets with identical capacity c into G (i.e., $c = |V|$). In other words, there is no capacity constraint on each

cloudlet. We can see that an optimal solution to the capacitated cloudlet placement problem in G also is an optimal solution to the metric K-median problem in G'. Since the metric K-median problem is NP-hard [6], the capacitated cloudlet placement problem is NP-hard too. □

4 Algorithm for the Capacitated Cloudlet Placement Problem

In this section we first provide an integer linear program solution, and then devise a fast, scalable heuristic for the capacitated cloudlet placement problem.

4.1 Integer Linear Programming Formulation

We here formulate the capacitated cloudlet placement problem as an ILP. We first define a set of decision variables. Recall that, in the capacitated cloudlet placement problem, K cloudlets need to be placed into K different locations in a set S of potential locations in a WMAN $G = (V \cup S, E)$, such that the average access delay of user requests is minimized, while meeting the computing demands of each user request. This is equivalent to decide which cloudlets are placed to which locations in S, and which requests are assigned to which cloudlets for processing. We thus use a binary decision variable p_{il} to indicate whether cloudlet C_i will be placed to location $v_l \in S$, where $p_{il} = 1$ if cloudlet C_i is placed at v_l; $p_{il} = 0$ otherwise, for all i and l with $1 \leq i \leq K$ and $1 \leq l \leq |S|$. Similarly, we use an indicator decision variable x_{lm_j} to indicate whether a user request $r_m \in \mathcal{R}_j$ will be assigned to a cloudlet located at v_l for processing. That is, $x_{lm_j} = 1$ if $r_m \in \mathcal{R}_j$ is assigned to the cloudlet at location v_l; and 0 otherwise. Let z_{jl} be the number of user requests at AP v_j that are assigned to the cloudlet at location v_l. Clearly, $\sum_{r_m \in \mathcal{R}_j} x_{lm_j} = z_{jl}$. Notice that, different user requests in $\mathcal{R}_j$ may be assigned to different cloudlets for processing. We thus have $0 \leq z_{jl} \leq w(v_j)$ and $\sum_{l=1}^{|S|} z_{jl} = w(v_j)$, where $w(v_j)$ is the number of requests at AP v_j, i.e., $w(v_j) = |\mathcal{R}_j|$.

Recall that d_{jl} is the length of a shortest path between $v_j \in V$ and $v_l \in S$ in terms of the accumulated delay of its edge delays. The objective of the capacitated cloudlet placement problem is to minimize the average cloud access delay of the requests of all APs in V, i.e.,

$$minimize \quad \frac{\sum_{j=1}^{|V|} \sum_{l=1}^{|S|} z_{jl} d_{jl}}{\sum_{j=1}^{|V|} w(v_j)}, \tag{1}$$

subject to the following constraints

$$\sum_{l=1}^{|S|} p_{il} = 1 \quad \text{for each cloudlet } C_i, 1 \leq i \leq K, \tag{2}$$

$$\sum_{i=1}^{K} p_{il} \leq 1 \quad \text{for each location } v_l \in S, \tag{3}$$

$$\sum_{r_m \in \mathcal{R}_j} x_{lm_j} = z_{jl} \quad \text{for each AP} v_j \in V, \text{ and each location } v_l \in S, \tag{4}$$

$$\sum_{l=1}^{|S|} z_{jl} = w(v_j) \quad \text{for each AP} v_j \in V, \tag{5}$$

$$\frac{z_{jl}}{w(v_j)} \leq \sum_{i=1}^{K} p_{il} \quad \text{for each } v_j \in V \text{ and } v_l \in S, \tag{6}$$

$$\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \gamma_m \cdot x_{lm_j} \leq \sum_{i=1}^{K} p_{il} c_i \quad \text{for each } v_l \in S, \tag{7}$$

$$p_{il} \in \{0, 1\}, \tag{8}$$

$$x_{lm_j} \in \{0, 1\}, \tag{9}$$

$$z_{jl} \in \{0, 1, \ldots, w(v_j)\}, \tag{10}$$

where constraint (2) ensures that each of the K cloudlets is placed at only one location in S, and constraint (3) ensures that at most one cloudlet is placed at any location $v_l \in S$. Constraints (4) and (5) jointly ensure that all user requests from each AP v_j will be assigned to cloudlets for processing, while constraint (6) ensures that whenever some user requests at AP v_j are assigned to location v_l, then, one of the K cloudlets must be placed at location v_l. Constraint (7) ensures that the total allocated computing resources to the user requests processed by each cloudlet is no more than its capacity.

4.2 Heuristic Algorithm

The ILP solution is only applicable when the problem size is small; otherwise, it suffers from poor scalability. We here develop a fast, scalable solution as follows. We assume that the K cloudlets have been sorted in decreasing order of their capacities, i.e., cloudlet C_i has a capacity c_i with $c_1 \geq c_2 \geq \ldots \geq c_K$.

To minimize the average cloudlet access delay without violating computing capacities of cloudlets, each cloudlet should be placed to a location that can 'cover' as many user requests as possible. The heuristic proceeds iteratively to find the next location for the next cloudlet placement. Within iteration i, assume that cloudlets $C_1, C_2, \ldots, C_{i-1}$ have already been placed to locations $v_1, v_2, \ldots, v_{i-1}$, respectively. Cloudlet C_i then will be placed to a location v_i that have not been occupied by any placed cloudlet, i.e., $v_i \in U_i = S \setminus \{v_1, v_2, \ldots, v_{i-1}\}$. To find such a location v_i, for each potential location $v_l \in U_i$, the minimum sum D_{il} of delays of the user requests assigned to cloudlet C_i is calculated if C_i is placed at location v_l. Cloudlet C_i will be placed to a location with the minimum value of D_{il}s, i.e., $v_i = \arg\min_{v_l \in U_i}\{D_{il}\}$. The calculation of D_{il} for each location v_l is as follows.

The APs that have requests to be assigned are first sorted in increasing order of delays between the APs and a potential location v_l of cloudlet C_i, i.e., d_{lj} for each such an AP v_j. The first least r APs in the sorted AP sequence with the sum of the resource demands by their remaining user requests being no less than the capacity c_i of C_i is identified. All the user requests from the first $r-1$ APs and some of the user requests from the rth AP will be assigned to C_i. In particular, to determine which user requests at the rth AP should be assigned to C_i for processing, all user requests at the rth AP are sorted in increasing order of their computing resource demands. The sorted user requests are assigned to C_i one by one until the capacity c_i of C_i reaches. As a result, D_{il} is the sum of delays of the user requests assigned to cloudlet C_i at potential location v_l. The detailed algorithm is described in `Algorithm` 1.

Algorithm 1. `Greedy_Heuristic`

Input: A WMAN $G(V \cup S, E)$, the set of user requests $\mathcal{R}_j$ at each AP $v_j \in V$ and the computing resource demand γ_m of each user request $r_m \in \mathcal{R}_j$ with $w(v_j) = |\mathcal{R}_j|$, the communication delay $d(e)$ of each edge $e \in E$, K cloudlets with capacities $c_1, c_2, \ldots,$ and c_K, respectively, and the set S ($\subseteq V$) of potential locations for the K cloudlets.
Output: The placement locations of the K cloudlets.

$L \leftarrow \emptyset$; /* the set of placed cloudlet locations */
Compute all pairs of shortest paths for each pair of APs in G;
Sort the K cloudlets by their capacities in decreasing order. Cloudlet C_i has a capacity c_i and $c_1 \geq c_2 \geq \ldots \geq c_K$;
for $i \leftarrow 1$ to K **do**
  /*Place cloudlet C_i with capacity c_i to an unoccupied location*/
  $U_i \leftarrow S \setminus L$; / * the set of potential locations */
  for each potential location $v_l \in U_i$ **do**
    Sort the APs with to-be-allocated user requests in increasing order of their access delays d_{lj} between AP v_j and location v_l of cloudlet C_i;
    Find the first r APs in the sorted AP sequence such that the sum of the resource demands of the user requests in these APs is no less than the computing capacity c_i of C_i.
    Assign the user requests from the first $r-1$ APs to cloudlet C_i at location v_l;
    Sort the requests at the rth AP in increasing order of their computing resource demands;
    Allocate a subset of user requests at the rth AP to cloudlet C_i until its computing capacity c_i is met;
    Let D_{il} be the sum of delays of the requests assigned to the cloudlet C_i at location v_l;
  end for
  Place cloudlet C_i at the location v_i with the minimum sum of delays, i.e., $v_i = \arg\min_{v_l \in U_i}\{D_{il}\}$;
  $L \leftarrow L \cup \{v_i\}$;
end for
return L.

Theorem 1. *Given a WMAN $G = (V \cup S, E)$, a set $\mathcal{R}_j$ of user requests at each AP $v_j \in V$, a set of user requests $\mathcal{R}_j$ at each v_j with $w(v_j) = |\mathcal{R}_j|$, and K cloudlets $C_1, C_2, \ldots, C_K$ with capacities $c_1, c_2, \ldots, c_K$, respectively, there is a fast, scalable algorithm for the capacitated cloudlet placement problem, which takes $O(Kn^2 \log n + nm)$ time, assuming that $w(v) \leq \min_{1 \leq i \leq K}\{c_i\}$ for any $v \in V$ and $1 \leq K \leq |S| \leq |V|$, where $n = |V|$ and $m = |E|$.*

Proof. It can be seen that the solution delivered by `Algorithm` 1 is a feasible solution since all user requests at each AP $v_j \in V$ are assigned to cloudlets and the number of user requests assigned to each cloudlet C_i is no more then its capacity c_i. In the following we analyze the time complexity of `Algorithm` 1.

Let $n_1 = |V| + |S|$ and $m = |E|$ be the number of nodes and edges in G. Finding all pairs of shortest paths in G takes time $O(n_1^2 \log n_1 + n_1 m) = O(n^2 \log n + nm)$, by applying Dijkstra's algorithm for single-source shortest paths for all source nodes in $V \cup S$, where the time complexity of Dijkstra's algorithm is $O(n_1 \log n_1 + m \log n_1)$ [9]. `Algorithm` 1 proceeds iteratively and one of the K cloudlets will be placed within each iteration. When it places cloudlet C_i within iteration i, identifying a location $v_i \in S \setminus \{v_1, v_2, \ldots, v_{i-1}\}$ for cloudlet C_i placement takes $O(K \cdot |S| \cdot |V| \log |V|) = O(Kn^2 \log n)$ time as $|S| \leq |V| = n$. The time complexity of `Algorithm` 1 thus is $O(n^2 \log n + nm + Kn^2 \log n) = O(Kn^2 \log n + nm)$. □

5 Approximation Algorithms with Identical Cloudlet Capacities

In this section, we deal with a special case of the problem where all K cloudlets have identical capacities c and devise two approximation algorithms with guaranteed approximation ratios depending on whether all user requests have identical computing resource demands or not. We start with each user request having identical resource demands, we then show how to extend this solution to the general case where different user requests may have different resource demands.

5.1 An Approximation Algorithm with Identical Resource Demands

For simplicity, we assume that each request has one unit computing resource demand (e.g., one virtual machine (VM)) [30]. That is, $\gamma_m = 1$ for each user request $r_m \in \mathcal{R}_j$ at each AP v_j. Even for this special case, the problem is still NP-hard, which can be reduced from *the capacitated K-median problem*, by assuming that each AP has only one user request. Intuitively, this problem can be solved by applying an approximation algorithm for the capacitated K-median problem [6] in a graph, where the graph is derived from G by replacing each AP node v_j in G with $w(v_j)$ virtual AP nodes and each virtual AP node has a single user request. However, finding a solution in such a large-scale graph is painstaking, since each AP may have a large number of user requests (e.g., the network size is several hundreds, while the number of mobile users can be over millions). In the following we will devise a fast yet scalable approximation algorithm for the problem by adopting a novel scaling technique.

The capacitated K-median problem. Given a set of locations U with each location $j \in U$ having a demand $w_j \geq 0$, K centers with identical service capacity M, and the service cost d_{ij} for a center at location i serving one unit demand from location j, *the capacitated K-median problem* is to find K different locations in U to place the K centers and allocate the demand w_j of each location $j \in U$ to one of the K placed centers such that the total service cost is minimized, subject to the capacity constraint on each center M and demand service constraint that the demand from a location $j \in U$ must be served by only one center, where the service costs are non-negative, symmetric, and satisfy the triangle inequality.

Theorem 2. *[6] There is an approximation algorithm for the capacitated K-median problem in the metric space, which delivers an approximate solution with an approximation ratio 16 in the service cost, while the total demand served by each center is no more than four times of its capacity M.*

We now devise an approximation algorithm for this special capacitated cloudlet placement problem with identical resource demands. Our strategy is to reduce the problem

to the capacitated K-median problem, an approximate solution to the latter will form a base of the approximate solution to the former through a proper transformation. According to Theorem 2 that if the number of user requests from all APs is polynomial of the number of APs n in the WMAN, this special case of the capacitated cloudlet placement problem can be reduced to the capacitated K-median problem, by replacing the $w(v_j)$ user requests at each AP v_j with $w(v_j)$ virtual nodes, and each virtual node has only one user request. An approximate solution to the latter in turn returns an approximate solution to the former. However, the number of users in a WMAN typically is several orders of magnitudes of the network size n, which makes the user request assignment become a painstaking job. In the following we propose a scaling technique to speed up the user request assignment to cloudlets.

Let N_0 be a basic unit (a block) of the number of user requests to be allocated to a cloudlet, where N_0 is an integer with $1 \leq N_0 \leq \min_{v_j \in V}\{w(v_j)\}$, and we further assume that $N_0 \leq c$. Thus, each AP $v_j \in V$ will have $n_j = \lfloor w(v_j)/N_0 \rfloor$ units. Similarly, each cloudlet C_i will have a capacity of $c' = \lceil c/N_0 \rceil$ units. Given the original graph $G = (V, E)$, the number of user requests at each AP $w : V \mapsto \mathbb{N}$, the data transmission delay between every two APs $d : E \mapsto \mathbb{R}^{\geq 0}$, and the K cloudlets with identical capacity c, an auxiliary complete graph $G_U = (V_U, E_U)$ is constructed as follows. For each AP v_j with $w(v_j)$ user requests, $n_j = \lfloor w(v_j)/N_0 \rfloor$ virtual nodes $v_j^1, v_j^2, \ldots, v_j^{n_j}$ are added to V_U with each as a source node of demand 1. There is an edge e in E_U for each pair of virtual nodes v_i^x and v_j^y in V_U. The weight of edge $e = (v_i^x, v_j^y) \in E_U$ is the accumulated delay between APs v_i and v_j in G of N_0 user requests (i.e., $N_0 d_{ij}$) between them if $v_i \neq v_j$, and 0 otherwise. The problem then is to place the K cloudlets with identical capacities c' in V_U to cover all source nodes such that the weighted sum of the edges between the placed cloudlets and their covered source nodes is minimized. This is *the capacitated K-median problem*, which can be solved by an approximation algorithm due to Charikar et al. [6].

Having the approximate solution, a feasible solution to the original problem then can be obtained. That is, the K cloudlets will be placed at the K locations provided by the approximate solution. For each AP v_j with $n_j = \lfloor w(v_j)/N_0 \rfloor$ virtual nodes $v_j^1, v_j^2, \ldots, v_j^{n_j}$ in G_U, assume that v_j^l is allocated to a cloudlet at location i in the approximate solution, then N_0 user requests at AP v_j will be assigned to the cloudlet, where $1 \leq l \leq n_j$. The remaining $w(v_j) - N_0 \cdot n_j$ ($< N_0$) user requests at AP v_j will be assigned to a nearest cloudlet with the minimum accumulated access delay among the cloudlets at which the virtual nodes $v_j^1, v_j^2, \ldots, v_j^{n_j}$ are allocated. The detailed algorithm is given in `Algorithm` 2.

Algorithm 2. `Appro`

Input: a WMAN $G(V, E)$, the number $w(v_j)$ of user requests at each AP $v_j \in V$, the delay $d(e)$ of each edge $e \in E$, K cloudlets with each having a capacity c, and a positive integer N_0 with $1 \leq N_0 \leq \min_{v_j \in V}\{w(v_j)\}$ and $N_0 \leq c$.

Output: The placement locations of the K cloudlets.

1: Construct an auxiliary graph $G_U = (V_U, E_U)$ from G;
2: Find an approximate solution to the capacitated K-median problem in $G_U(V_U, E_U)$, by applying the algorithm due to Charikar et al. [6];
3: Place the K cloudlets to their corresponding locations in the found approximate solution;
4: For each virtual node v_j^l derived from AP v_j, assign N_0 user requests at AP v_j to the cloudlet at which v_j^l is allocated in the approximate solution;
5: Assign the rest of $w(v_j) - N_0 n_j$ user requests at AP v_j to the nearest cloudlet at which the n_j virtual nodes of AP v_j are allocated.

5.2 Approximation Algorithm with Different Resource Demands

We now consider another special capacitated cloudlet placement problem where all cloudlets have identical capacities but different user requests may have different resource demands, and how to extend algorithm `Appro` for this special capacitated cloudlet placement problem.

Recall that the computing resource demand of each request $r_m \in \mathcal{R}_j$ of AP v_j is γ_m. We modify the proposed approximation algorithm `Appro` for this general case of different resource demands. Let γ_{max} and γ_{min} be the maximum and minimum amounts of computing resource demands among user requests, respectively. Without loss of generality, we assume that $\rho = \frac{\gamma_{max}}{\gamma_{min}}$ is a given constant and the computing resource demand γ_m of each request r_m is divisible by γ_{min}. Specifically, we treat each user request r_m as ρ_m ($= \frac{\gamma_m}{\gamma_{min}}$) *virtual user requests* $r'_{m1}, r'_{m2}, \ldots, r'_{m\rho_m}$ with each having identical demands γ_{min}. We then apply algorithm `Appro` for all virtual user requests with identical computing resource demands. That is, all virtual user requests at each AP v_j will be divided into a number of blocks, and each block contains exactly N_0 virtual user requests. The solution delivered may not be a feasible solution to the original problem, since the computing resource demands of a user request may be split into different cloudlets for processing. To obtain a feasible solution, we modify the solution by merging the virtual user requests derived from a user request to one of their allocated cloudlets. We distinguish the merge into three cases.

Case 1: All the virtual user requests of a given user request r_m are contained in a single block, which means that all virtual user requests in that block will be assigned to a single cloudlet for processing. Nothing needs to be adjusted in this case.

Case 2: The virtual user requests of r_m are contained in multiple blocks, and all these blocks are assigned to a single cloudlet. In other words, all virtual user requests of r_m are processed by a single cloudlet. We thus do nothing in this case.

Case 3: The virtual user requests of r_m are contained in multiple blocks, and these blocks are assigned to different cloudlets. In this case, we adjust the allocations of virtual user requests to ensure that all virtual user requests of r_m are allocated to one cloudlet as follows. Assume that the virtual user requests of r_m are contained in blocks $b_{m1}, \ldots b_{ml}, \ldots, b_{mL}$, and all virtual user requests in block b_{ml} is assigned to cloudlet C_l with $1 \leq l \leq L$ and $1 \leq l \leq L \leq K$. We further assume that cloudlet C_{l_0} is the cloudlet that

results in the maximum access delay $\tau_{ml_0}^{max}$ by some virtual requests of user request r_m, where $\tau_{ml_0}^{max} = \max\{\tau_{ml} \mid 1 \leq l \leq L\}$. We then merge the virtual user requests of r_m from all other cloudlets and assign them to C_{l_0}. The details of the proposed algorithm is described in Algorithm 3.

Algorithm 3. Appro-Extension

Input: a WMAN $G(V, E)$, a set $\mathcal{R}_j$ of user requests at each AP $v_j \in V$, the number $w(v_j)$ of user requests in $\mathcal{R}_j$, the computing resource demand γ_m for each user request $r_m \in \mathcal{R}_j$, the delay $d(e)$ of each edge $e \in E$, K cloudlets with a uniform computing capacity c, and a positive integer N_0 with $1 \leq N_0 \leq \min_{v_j \in V}\{w(v_j)\}$ and $N_0 \leq c$.

Output: The placement locations of the K cloudlets.

1: Let γ_{min} and γ_{max} be the minimum and maximum resource demands of a single user request at any AP in V;
2: **for** each AP $v_j \in V$ **do**
3: **for** each of user request $r_m \in \mathcal{R}_j$ **do**
4: Construct $\frac{\gamma_m}{\gamma_{min}}$ virtual user requests with each having the computing resource demand γ_{min}, assuming that γ_m is divisible by γ_{min};
5: **end for**
6: **end for**
7: Find a solution for the K cloudlets by invoking algorithm Appro, where each AP v_j contains $\sum_{r_m \in \mathcal{R}_j} \frac{\gamma_m}{\gamma_{min}}$ virtual user requests with each having computing resource demand γ_{min};
8: **for** each AP $v_j \in V$ **do**
9: Let $b_{m1}, b_{m2}, \ldots, b_{ml}, \ldots, b_{mL}$ the set of blocks that contain the virtual requests of r_m, and each block has been assigned to a cloudlet in the found solution;
10: /* merge all virtual user requests of a user request from different cloudlets to the cloudlet with the maximum access delay from the AP in which the request is */;
11: **if** any of these L blocks is assigned to a different cloudlet **then**
12: Identify the cloudlet C_{l_0} with the maximum access delay in processing the virtual user requests of r_m, i.e., $\tau_{ml_0}^{max} = \max\{\tau_{ml} \mid 1 \leq l \leq L\}$, where τ_{ml} is the delay between the virtual requests in block b_{ml} and the cloudlet to which they will be allocated;
13: Merge all virtual user requests that were allocated to other cloudlets to cloudlet C_{l_0}.
14: **end if**
15: **end for**

5.3 Algorithm Analysis

The rest is to analyze the approximation ratios of the two proposed algorithms Appro and Appro-Extension as follows.

We start by analyzing the approximation ratio and time complexity of algorithm Appro. Let $G' = (V', E'; d')$ be a graph that is identical to the original graph $G = (V, E; d)$, i.e., $V' = V$, $E' = E$, and $d'_{ij} = d_{ij}$ for any two APs v_i and v_j in V. However, assume that the number of user requests at each AP $v_{j'}$ in G' is $\lfloor w(v_j)/N_0 \rfloor N_0$, which is no more than the number of user requests $w(v_j)$ at AP v_j in G, i.e., $\lfloor w(v_j)/N_0 \rfloor N_0 \leq w(v_j)$. We then deploy K cloudlets in network G' while the capacity of each cloudlet C_i now is set at $\lceil \frac{c}{N_0} \rceil N_0$, not c. It is obvious that $\lceil \frac{c}{N_0} \rceil N_0 \geq c$. For each AP $v_{j'}$ in G', we can split its $\lfloor w(v_j)/N_0 \rfloor N_0$ user requests into $n_j = \lfloor w(v_j)/N_0 \rfloor$ blocks with each block containing exactly N_0 user requests. We consider two types of allocations of user requests blocks: one is that each user request from a block is allowed to be allocated to any cloudlet; and another is that the whole N_0 user requests of each block must be allocated to a single cloudlet but the user requests in different blocks can be allocated to different cloudlets. In the following we show that the costs of the optimal solutions to these two types of request allocations are equal.

Theorem 3. *Denote by OPT_1 and OPT_2 the costs of the optimal solutions to these two types of the special cloudlet placement problem in graph $G' = (V', E'; d')$ respectively, we have $OPT_1 = OPT_2$.*

Proof. It can be seen that $OPT_1 \leq OPT_2$, since the optimal solution to the second type is a feasible solution to the first type one. The rest is to show that $OPT_2 \leq OPT_1$. Denote by $\mathbb{X}_1$ and $\mathbb{X}_2$ the optimal solutions to the first and second types of request allocations, respectively. Assume that the K cloudlets are placed at locations $i_1, i_2, \ldots, i_K$ in solution $\mathbb{X}_1$. Recall that the $\lfloor w(v_j)/N_0 \rfloor N_0$ user requests at AP $v_{j'}$ have been split into $n_j = \lfloor w(v_j)/N_0 \rfloor$ blocks with each block containing N_0 requests exactly. Let $n_r = \sum_{v_j \in V} n_j$ be the number of blocks in G' that can be represented by $b_1, b_2, \ldots, b_{n_r}$, respectively. Let $B = \{b_1, b_2, \ldots, b_{n_r}\}$ be the set of n_r blocks and $I = \{i_1, i_2, \ldots, i_K\}$ the set of the K cloudlet locations in solution $\mathbb{X}_1$.

We construct an auxiliary flow graph $G_f = (\{s\} \cup B \cup I \cup \{t\}, E_f)$ from the n_r blocks in B and the K locations in I as follows. There is a directed edge in E_f from source s to each block $b_j \in B$ with capacity 1 and cost $d''_{sj} = 0$. For each block $b_j \in B$ and each location $i_l \in I$, there is a directed edge in E_f from b_j to i_l with capacity 1 and cost $d''_{jl} = d_{jl} N_0$ (i.e., the total delay of transmitting the N_0 user requests from block b_j to the cloudlet located at i_l). Furthermore, there is an edge in E_f from each location $i_l \in I$ to sink t with capacity $\lceil \frac{c}{N_0} \rceil$ units and cost $d''_{lt} = 0$.

Given a flow f from s to t in graph G_f, the cost of the flow is defined as $\sum_{e \in E_f} f_e \cdot d''_e$. Following the assumption that the number of user requests in G is no more than the total capacity of the K cloudlets, i.e., $\sum_{v_j \in V} w(v_j) \leq K \cdot c$, we then have $n_r = \sum_{v_j \in V} \lfloor w(v_j)/N_0 \rfloor \leq K \lceil \frac{c}{N_0} \rceil$, and the value of a maximum flow in G_f from s to t is n_r.

The minimum cost maximum flow problem in G_f is to find a maximum flow from node s to node t with the minimum cost [1]. From the optimal solution $\mathbb{X}_1$ to the first type of request allocation, we can construct a fractional maximum flow f (not necessarily having the minimum cost of the flow) to the minimum cost maximum flow problem in G_f, i.e., for the N_0 user requests in each block b_j, assume that x_{ji} user requests of the N_0 requests are allocated to the cloudlet located at $i \in I$, the fractional flow f_{ji} from block b_j to location i is $\frac{x_{ji}}{N_0}$. It can be easily verified that the cost of this fractional flow is OPT_1. On the other hand, the capacity of each edge in graph G_f is integral. Following the well-known integrality property for the minimum cost maximum flow problem [1], there

is an integral minimum cost maximum flow f^* for the problem. That is, for each block $b_j \in B$ and each location $i_l \in I$, the flow f^*_{jl} from b_j to i_l is either 0 or 1 as the capacity of edge (b_j, i_l) is 1. Denote by $D(f^*)$ the cost of flow f^*, i.e., $D(f^*) = \sum_{e \in E_f} f^*_e \cdot d''_e$. Then, $D(f^*) \leq OPT_1$, since the solution $\mathbb{X}_1$ is a feasible solution to the minimum cost maximum flow problem in G_f. Also, we know that this integral maximum flow f^* corresponds to a feasible solution with cost $D(f^*)$ to the second type of request allocation. As $\mathbb{X}_2$ with cost OPT_2 is an optimal solution to the problem, then $OPT_2 \leq D(f^*)$. Therefore, we have $OPT_2 \leq D(f^*) \leq OPT_1$. We thus have $OPT_2 = OPT_1$. □

Theorem 4. *Given a WMAN $G = (V, E)$ with $w(v_j)$ user requests at each AP $v_j \in V$ and K cloudlets with identical capacity of c, assume that each request takes a unit of computing resource. There is an approximation algorithm, `Algorithm 2`, for the special cloudlet placement problem with identical resource demands, which delivers a solution with the average access delay of $16(1+\epsilon)$-optimal, while the accumulated computing demand by each cloudlet is no more than $8(1+\delta)c$. The time complexity of the algorithm is $O((\sum_{j=1}^{n} n_j)^9)$, where $\epsilon = \max_{v_j \in V}\{\frac{1}{\lfloor w(v_j)/N_0 \rfloor}\} \leq 1$, $\delta = \frac{N_0}{c} \leq 1$, N_0 is a positive integer with $1 \leq N_0 \leq \min_{v_j \in V}\{w(v_j)\}$, $N_0 \leq c$, $n_j = \lfloor \frac{w(v_j)}{N_0} \rfloor$ for all j with $1 \leq j \leq n$.*

Proof. We first analyze the approximation ratio of `Algorithm 2` as follows. Denote by OPT the optimal (or minimum) total delay for the special capacitated cloudlet placement problem in $G(V, E)$. Denote by OPT_U the optimal total service cost for the capacitated K-median problem in the auxiliary graph $G_U(V_U, E_U)$.

Following the construction of G_U and Theorem 3, OPT_U is no more than OPT as the number of user requests $n_j N_0$ at each AP v_j in G_U is no more than the number of user requests $w(v_j)$ at AP v_j in G, i.e., $\lfloor w(v_j)/N_0 \rfloor N_0 \leq w(v_j)$, and the capacity $\lceil \frac{c}{N_0} \rceil N_0$ of each cloudlet in G_U is no less than the capacity c of the cloudlet in G, i.e., $\lceil \frac{c}{N_0} \rceil N_0 \geq c$. Following Theorem 2, the cost (or the total delay) of the approximate solution delivered by the approximation algorithm for the capacitated K-median problem in graph G_U is no more than $16 \cdot OPT_U$. Denote by D_j the total delay incurred by assigning the $n_j N_0$ user requests at AP v_j to their allocated cloudlets in the approximate solution. Then, $\sum_{v_j \in V} D_j \leq 16 \cdot OPT_U$. Since the rest $w(v_j) - n_j N_0$ ($\leq N_0$) user requests at AP v_j are assigned to their nearest cloudlet among the cloudlets to which the $n_j N_0$ user requests are allocated, the total delay incurred by these $w(v_j) - n_j N_0$ user requests is no more than $\frac{D_j}{n_j N_0} N_0 = \frac{D_j}{n_j}$. Therefore, the total delay by assigning all user requests to the K cloudlets is no more than $\sum_{v_j \in V}(D_j + \frac{D_j}{n_j}) = \sum_{v_j \in V} D_j(1 + \frac{1}{n_j}) \leq (1+\epsilon)\sum_{v_j \in V} D_j \leq (1+\epsilon)16 \cdot OPT_U \leq (1+\epsilon)16 \cdot OPT$, where $\epsilon = \max_{v_j \in V}\{\frac{1}{n_j}\} = \max_{v_j \in V}\{\frac{1}{\lfloor w(v_j)/N_0 \rfloor}\} \leq 1$.

We then show that the number of user requests served by each cloudlet in the solution by `Algorithm 2` is no more than $8(1+\delta)c$, where $\delta = \frac{N_0}{c} \leq 1$. Following Theorem 2, the number of user requests allocated to each cloudlet is no more than $4\lceil \frac{c}{N_0} \rceil N_0 \leq 4(\frac{c}{N_0} + 1)N_0 = 4(1 + \frac{N_0}{c})c = 4(1+\delta)c$ in the approximate solution delivered by the approximation algorithm in [6]. We show that after having assigned the rest $w(v_j) - n_j N_0$ ($\leq N_0$) user requests from each AP v_j to the cloudlets, the number of user requests served by each cloudlet is no more than twice the number of user requests prior to this assignment. Assume that for a deployed cloudlet C_i, it processes user requests from APs $v_1, v_2, \ldots, v_p$ before assigning the remaining user requests. Following `Algorithm 2`, cloudlet C_i will process no less than N_0 user requests from each of these p APs. Since there are no more than N_0 remaining user requests at each of the p APs, the number of user requests assigned to cloudlet C_i thus is no more than $8(1+\delta)c$.

We finally analyze the time complexity of `Algorithm 2`. It can be seen that $G_U = (V_U, E_U)$ contains $n_U = \sum_{j=1}^{n} n_j$ nodes and $m_U = \frac{n_U(n_U-1)}{2}$ edges, where $n_j = \lfloor \frac{w(v_j)}{N_0} \rfloor$ and $1 \leq j \leq n$. The time complexity of `Algorithm 2` is dominated by invoking the approximation algorithm for the capacitated K-median problem in G_U [6], while the running time of the latter is dominated by the time of finding an optimal solution to the liner programming relaxation of the capacitated K-median problem. Given an LP, it is shown that there is an algorithm with time complexity $O(n_v^{3.5} L)$ for calculating its optimal solution, where n_v is the number of variables in the LP and L is the number of bits in the input of the LP [17]. Following [6], there are $n_v = n_U + m_U$ variables in the LP of the capacitated K-median problem and the number of bits in the input L is $O(n_U + m_U)$. Therefore, the time complexity of `Algorithm 2` is $O(n_v^{3.5} L) = O((n_U + m_U)^{3.5}(n_U + m_U)) = O((n_U + m_U)^{4.5}) = O(m_U^{4.5}) = O(n_U^9) = O((\sum_{j=1}^{n} n_j)^9)$. □

Notice that if the capacity of each cloudlet is not allowed to be overloaded, we may set the capacity of each cloudlet to $\frac{c}{8(1+\delta)}$ instead of c, then apply the approximation algorithm for this new capacity. Following Theorem 4, none of the cloudlet will violate its original capacity c.

We proceed by analyzing the performance of algorithm `Appro-Extension` by the following theorem.

Theorem 5. *Given a WMAN $G = (V, E)$, K cloudlets with identical capacity of c, a set $\mathcal{R}_j$ of user requests at each AP $v_j \in V$ with different computing resource demands, let $w(v_j) = |\mathcal{R}_j|$, assuming that the maximum and minimum resource demands among user requests are γ_{max} and γ_{min} respectively, and the computing resource demand γ_m of any user request r_m is divisible by γ_{min}. Then, there is an approximation algorithm, algorithm `Appro-Extension`, for the special cloudlet placement problem with different resource demands, which delivers a solution with the approximation ratio of $16\rho(1+\epsilon)$ while the total computing resource needed by each cloudlet is no more than $8\rho(1+\delta)$ times of its optimal one c. The time complexity of the algorithm is $O((\sum_{j=1}^{n} n_j)^9)$, where $\rho = \frac{\gamma_{max}}{\gamma_{min}}$ is an integer, $\epsilon = \max_{v_j \in V}\{\frac{1}{\lfloor w(v_j)/N_0 \rfloor}\} \leq 1$, $\delta = \frac{N_0}{c} \leq 1$, $N_0 \leq c$ is a positive integer with $1 \leq N_0 \leq \min_{v_j \in V}\{w(v_j)\}$, and $n_j = \lfloor \frac{w(v_j)}{N_0} \rfloor$ for all j with $1 \leq j \leq n$.*

Proof. We first analyze the approximation ratio of algorithm `Appro-Extension`. Let τ' be the average access delays delivered by algorithm `Appro` through treating each user request r_m as $\frac{\gamma_m}{\gamma_{min}}$ virtual user requests. Denote by τ the average access delay of user requests delivered by algorithm `Appro-Extension`. According to algorithm `Appro-Extension`, we can see that the difference between τ' and τ lies in the adjustments of virtual user requests of a user request that are assigned to different cloudlets in Case 3. Specifically, when the virtual user requests of r_m are allocated to multiple blocks (i.e., $b_{m1}, \ldots b_{ml}, \ldots b_{mL}$) and these blocks then are assigned to different cloudlets, all other virtual user requests of r_m in other blocks will be reassigned to cloudlet C_{l_0} that accommodates the virtual requests of r_m with the maximum access delay $\tau_{ml_0}^{max}$. Denote by τ'_{ml} the access delay that is experienced by one of the virtual user request in block b_{ml}. Clearly, we have $\sum_{l=1}^{L} \tau'_{ml} > \tau_{ml_0}^{max}$. Also, after moving all virtual user requests of r_m from other cloudlets to C_{l_0}, the delay τ_m experienced by user request r_m will be $\tau_{ml_0}^{max}$, i.e., $\tau_m = \tau_{ml_0}^{max}$. In addition, the number of virtual user requests moved to C_{l_0} is no more than ρ_m $(= \frac{\gamma_m}{\gamma_{min}})$ times of the number of virtual user requests in C_{l_0}. The relationship between τ' and τ is analyzed as follows:

$$\begin{aligned}
\tau' &= \frac{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \sum_{l=1}^{L} \tau'_{ml}}{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \frac{\gamma_m}{\gamma_{min}}} \\
&\geq \frac{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \tau_{ml_0}^{max}}{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \frac{\gamma_m}{\gamma_{min}}} \quad \text{since } \textstyle\sum_{l=1}^{L} \tau'_{ml} > \tau_{ml_0}^{max}, \\
&= \frac{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \tau_{ml_0}^{max}}{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \rho_m} \quad \text{since } \rho_m = \frac{\gamma_m}{\gamma_{min}}, \qquad (11) \\
&\geq \frac{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \tau_{ml_0}^{max}}{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \rho} \quad \text{since } \rho = \frac{\gamma_{max}}{\gamma_{min}} \geq \rho_m, \\
&= \frac{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \tau_{ml_0}^{max}}{\rho \sum_{j=1}^{n} w(v_j)} \\
&= \frac{\tau}{\rho}. \qquad (12)
\end{aligned}$$

Notice that the rationale from Eq. (11) to Eq. (12) is that $\tau_{ml_0}^{max}$ is the access delay of r_m by algorithm `Appro-Extension`. In other words, we have

$$\tau \leq \rho \tau', \qquad (13)$$

which means that the average delay will be no greater than ρ times of the average access delay τ' by the solution achieved through treating each user request r_m as ρ_m virtual user requests.

Let τ^* be the optimal solution to the capacitated cloudlet problem with identical cloudlet capacities and different user resource demands, and τ_m^* be the delay experienced by request r_m in the optimal solution τ^*. Similarly, denote by τ'^* the optimal solution of the problem by treating each user request r_m as $\frac{\gamma_m}{\gamma_{min}}$ virtual user requests, and τ'^*_{ml} the delay experienced by the virtual user request of user request r_m in block b_{ml} in this optimal solution. We now show that τ'^* is no greater than τ^* as follows:

$$\begin{aligned}
\tau'^* &= \frac{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \sum_{l=1}^{L} \tau'^*_{ml}}{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \frac{\gamma_m}{\gamma_{min}}} \qquad (14) \\
&\leq \frac{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \tau_m^*}{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \rho_m} \qquad (15) \\
&\leq \frac{\sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \tau_m^*}{\sum_{j=1}^{n} w(v_j)} \quad \text{since } \rho_m \geq 1, \qquad (16) \\
&= \tau^*,
\end{aligned}$$

where the rationale from Eq. (14) to Inequality (15) is that the total delay by all virtual user requests is a lower bound of the total delay of user requests, as the computing resource demands of a user request may be split into different cloudlets. Having inequalities (13), (16), and Theorem 4, we have

$$\tau \leq \rho \tau' \leq 16\rho(1+\epsilon)\tau'^* \leq 16\rho(1+\epsilon)\tau^*. \qquad (17)$$

Thus, τ will be no more than $16\rho(1+\epsilon)$ times of the optimal one τ^*. That is, the approximation ratio of algorithm `Appro-Extension` is $16\rho(1+\epsilon)$.

We then show the bound of the amount of computing resource allocated to user requests in each cloudlet. Since the number of virtual requests of request r_m that are moved to C_{l_0} is no more than ρ_m, the total computing resource demands in C_{l_0} will be no more than ρ_m times its current one by algorithm `Appro`. Thus, the total amount of computing resources allocated to user requests will be no more than ρ times of the capacity by algorithm `Appro`, i.e., $8\rho(1+\delta)c$.

The time complexity analysis is similar to the analysis in Theorem 4, omitted. □

6 Online User Request Assignment

In this section we deal with the dynamic user request assignment problem so as to minimize the average cloudlet access delay of mobile users, assuming that the K cloudlets have already been placed in the WMAN. We also assume that time is slotted into equal time slots, and user request assignments will proceed in the beginning of each time slot. In the following we first devise an online algorithm for such dynamic user request assignments, and then analyze the performance of the proposed online algorithm.

6.1 Online Algorithm

The idea behind the algorithm is similar to the one of Algorithm 3, the algorithm is invoked in the beginning of each time slot t. That is, we first create a number of virtual user requests for each user request at each AP, then reduce the virtual user request assignment problem to *the minimum-cost maximum flow problem* in another auxiliary flow network G', and finally adjust the virtual request assignments to ensure that each user request will be assigned to a single cloudlet for processing. The detailed algorithm is described as follows.

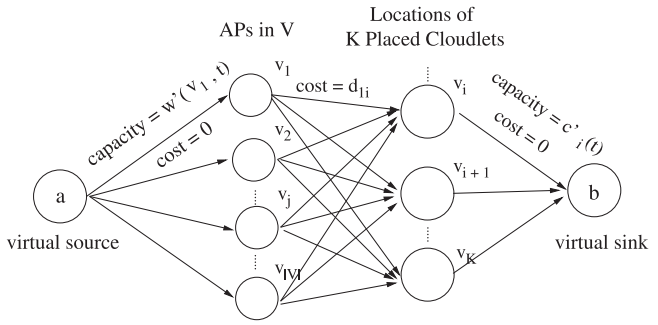


Fig. 2. The auxiliary graph $G' = (V \cup S \cup \{a, b\}, E')$.

For each request r_m at AP v_j, we first create $\rho_m = \frac{\gamma_m}{\gamma_{min}}$ *virtual user requests* $r'_{m,1}, r'_{m,2}, \ldots, r'_{m,\rho_m}$ with each having identical computing resource demand γ_{min}. We then construct a flow network $G' = (V \cup S \cup \{a, b\}, E')$ from the WMAN $G(V, E)$ and the K cloudlet placement locations, where there is a 'virtual source' a and a 'virtual sink' b in G'. For each AP $v_j \in V$, there is a directed edge in E' from source a to v_j with capacity $\sum_{r_m \in \mathcal{R}_j} \rho_m$ and a cost of 0, while $\mathcal{R}_j$ is the set of user requests at AP v_j. For each AP $v_j \in V$ and each cloudlet $v_i \in S$, there is a directed edge in E' from v_j to v_i with a sufficiently large capacity and a cost of d_{ji} (i.e., the shortest access delay of assigning a user request from AP v_j to cloudlet v_i). Furthermore, for each cloudlet located at $v_i \in S$, there is a directed edge in E' from v_i to sink b with capacity $\frac{c'_i(t)}{\gamma_{min}}$ (i.e., the maximum number of virtual user requests that can be processed by the cloudlet at location v_i with its residual capacity $c'_i(t)$ at time slot t) and cost 0. Fig. 2 shows the construction of such an auxiliary graph G'. Having G', the solution to the problem is to find a minimum-cost maximum flow f^* in G', by applying the algorithm in [1], which then returns a solution the dynamic user request assignment problem. Specifically, given the flow f^*, we can adjust the assignments of virtual user requests by merging the virtual user requests of each user request r_m that are assigned to different cloudlets to a single cloudlet. Assume that the virtual user requests $r'_{m,1}, r'_{m,2}, \ldots, r'_{m,\rho_m}$ of a user request r_m at AP v_j are assigned to a set of cloudlets $C_{i_1}, C_{i_2}, \ldots, C_{i_m}$ and the access delay from v_j to C_{i_k} is the minimum one, then, all these virtual requests will be re-assigned to cloudlet C_{i_k}. We refer to this online algorithm as Algorithm `Online_Assignment`.

6.2 Algorithm Analysis

We now analyze the correctness and the time complexity of the proposed online algorithm by Theorem 6 as follow.

Theorem 6. *There is an online algorithm* `Online_Assignment` *for the dynamic user request assignment problem in $G = (V, E)$, where the accumulated computing resource allocated to all assigned requests at each cloudlet C_i is no more than ρ times its residual computing capacity c'_i with $1 \leq i \leq K$, where $\rho = \frac{\gamma_{max}}{\gamma_{min}}$, γ_{max} and γ_{min} are the maximum and minimum resource demands by a single user request.*

Proof. It can be easily verified that the solution delivered by Algorithm `Online_Assignment` is a feasible solution, since each user request is assigned to a cloudlet. In the following, we show that the total computing resource demand by all newly assigned user requests at each cloudlet is no more than ρ times its residual capacity at that time slot.

In the assignment of virtual user requests in auxiliary graph G', none of the residual capacity of any cloudlet has been violated, as the capacity on edge $\langle v_i, b \rangle$ is the residual capacity of the cloudlet at location v_i. However, merging all virtual user requests of a user request from the other cloudlets to a cloudlet may violate the residual capacity constraint of the cloudlet to which all other virtual user requests move. Following Theorem 5, the number of virtual requests of each user request r_m is no more than ρ_m. Thus, the total residual capacity thus will be violated by no more than a factor of ρ as $\rho = \frac{\gamma_{max}}{\gamma_{min}}$. □

7 Performance Evaluation

In this section we evaluate the performance of the proposed algorithms, based on both real and synthetic network topologies.

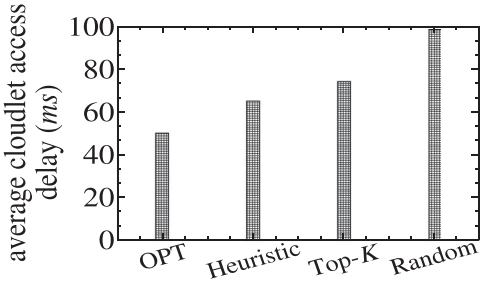
7.1 Experiment Settings

We assume that a WMAN $G(V, E)$ consists of 200 APs in the default setting, and there is an edge between every pair of nodes with a probability of 0.02. The network is generated by a popular tool GT-ITM [13]. We assume that the number of cloudlets is 10 percent of network size, i.e., $K = 10\% \cdot 200 = 20$, and the set of potential placement locations of the cloudlets is S with $S = V$. The number of user requests $w(v)$ at each AP $v \in V$ is randomly drawn from an interval [50, 500] [21]. To evaluate the proposed algorithms in real scenarios, we also adopt the real WIFI network topology in the Hong Kong Mass Transit Railway (HKMTR) [15], which contains 18 APs corresponding to the 18 districts in Hong Kong, and the number of requests at each AP is proportional to the population within its corresponding district. The computing resource demand γ_m of each user request r_m varies from 50 to 200 MHz [34], [35]. Let γ_{sum} be the total amount of computing resource demands of all user requests, then $\gamma_{sum} = \sum_{j=1}^{n} \sum_{r_m \in \mathcal{R}_j} \gamma_m$. The capacity of each cloudlet is randomly drawn from 100,000 MHz to $\alpha\gamma_{sum}$ MHz with $\alpha \geq 1$, and the sum of capacities of the K cloudlets is no less than γ_{sum}. The delay of each link in G is randomly generated between 5 and 50 ms [30]. Unless otherwise specified, these default parameters will be adopted in our simulation.

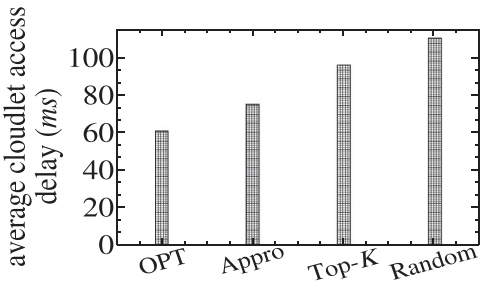
We evaluate the performance of proposed algorithms `Heuristic`, `Appro`, and `Appro-Extension` against other two heuristics. One places the K cloudlets to APs randomly. Another places the K cloudlets to the top-K APs, where an AP is a top-K AP if the number of user requests at it is one of the top-K values, and the rationale of this heuristic is to place cloudlets to K 'hotspot' (busiest) APs that have more user requests than others. For the sake of brevity, we refer to these two heuristics as algorithms `Random` and `Top-`K, respectively. In addition, the optimal solution obtained by the ILP is referred to as algorithm `OPT`, which can be found by applying the tool of lp_solve [24].

7.2 Performance Evaluation in the HKMTR Network

We first evaluate the performance of algorithms `Heuristic` and `Appro` against that of algorithms `Random`, `Top-`K, and



(a) Different cloudlet capacities and different resource demands



(b) Identical cloudlet capacities and identical resource demands

Fig. 3. The average cloudlet access delays of different algorithms in the HKMTR network.

OPT for the capacitated cloudlet placement problem, in the HKMTR network, by setting $K = 3$ and all the 18 AP locations being considered as the potential locations of the K cloudlets. Fig. 3a depicts the curves of the average cloudlet access delay delivered by algorithm Heuristic, from which it can be seen that algorithm Heuristic outperforms algorithms Random and Top-K by 30 and 10 percent in terms of the average cloudlet access delay, respectively. Furthermore, algorithm Heuristic delivers a near optimal average cloudlet access delay that is no more than 1.3 times of the optimal one. Fig. 3b shows the performance of approximation algorithm Appro against the other two benchmark algorithms, from which it can be seen that algorithm Appro significantly outperforms its benchmark counterparts.

7.3 Performance Evaluation of Different Algorithms in Synthetic Networks

We then evaluate the performance of different algorithms in synthetic networks, by varying the number of APs n from 10 to 1,000, while fixing the ratio of the number of cloudlets to the number of APs at 0.1, i.e., $K/n = 0.1$. Fig. 4 plots the curves of average cloudlet access delays delivered by different algorithms, from which it can be seen that algorithm

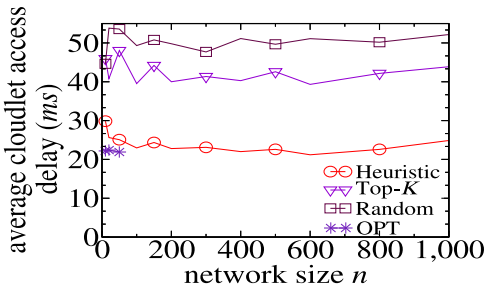


Fig. 4. The average cloudlet access delays of different algorithms with different cloudlet capacities and different resource demands.

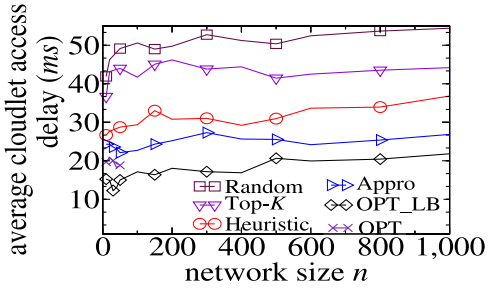


Fig. 5. The average cloudlet access delays of different algorithms with identical cloudlet capacities and identical resource demands.

Heuristic significantly outperforms algorithms Random and Top-K, and algorithm Top-K is only marginally better than algorithm Random. Specifically, the average cloudlet access delay by algorithm Heuristic is less than those by algorithms Random and Top-K by 25 and 35 percent, respectively. In addition, the figure also shows that the average cloudlet access delay by algorithm Heuristic is no more than 1.5 times of the optimal one, algorithm Heuristic thus is very promising in practice.

We now study the performance of approximation algorithm Appro against other heuristics for the problem when all cloudlets have identical capacities and all user requests have identical computing resource demands, i.e., $\gamma_m = 1$ for each user request r_m, by varying n from 10 to 1,000 while fixing $K/n = 0.1$. To evaluate the actual performance of algorithm Appro against its analytical result, we here use a lower bound on the optimal cost of the problem as an estimation of the optimal delay OPT as this is a minimization problem, where the lower bound is obtained by solving a linear programming that is relaxed from the ILP of the problem. It must be mentioned that this estimation to the optimal solution is very conservative, and the optimal solution OPT usually is much larger than this estimation. For the sake of convenience, we refer to OPT_LB as an estimation of OPT.

Fig. 5 plots the curves of average cloudlet access delays by algorithms Appro, Heuristic, Top-K, and Random against the lower bound OPT_LB of the optimal solution. It can be seen from this figure that the average cloudlet access delay by algorithm Appro is nearly optimal, which is only from 5 to 10 percent difference from the optimal one. Fig. 5 clearly indicates that the performance of algorithm Appro is significantly better than those of algorithms Heuristic, Top-K, and Random. Specifically, the average cloudlet access delay by algorithm Appro is 10 to 20 percent less than that by algorithm Heuristic, and nearly 50 percent less than those by algorithms Top-K and Random.

We thirdly study the performance of approximation algorithm Appro-Extension against other heuristics for the problem when all cloudlets have identical capacities but different users may request different amounts of computing resource, by varying n from 10 to 1,000 while fixing $K/n = 0.1$. It can be seen from Fig. 6 that the average cloudlet access delay by algorithm Appro-Extension is less than those of other algorithms, which is about 10, 40, and 50 percent lower than those of algorithms Heuristic, Top-K, and Random, respectively. In addition, the solution by algorithm Appro-Extension is nearly optimal, since its average cloudlet access delay is only around 10 percent

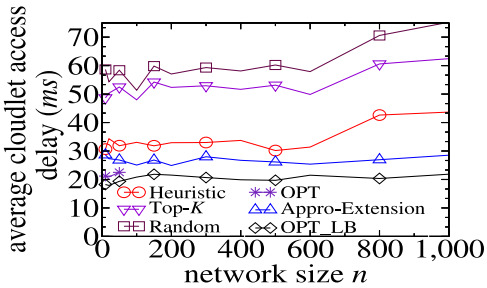


Fig. 6. The average cloudlet access delays of different algorithms with identical cloudlet capacities and different user resource demands.

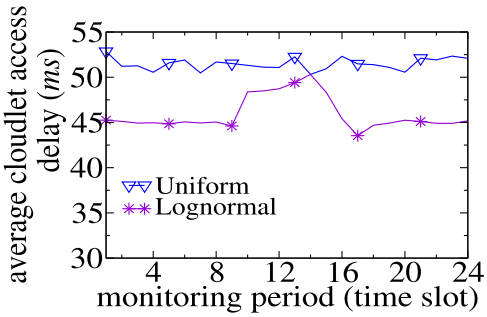


Fig. 8. The average cloudlet access delays of algorithm `Online_Assignment` under two different user request patterns: `Uniform` and `Lognormal`.

higher than that of `OPT` if $n \leq 50$, and 15 percent higher than that of `OPT_LB` otherwise. Further, the average cloudlet access delays by algorithms `Heuristic`, `Appro-Extension`, and `Random` rise slightly when n increases from 600 to 1,000, because in large-scale networks user requests usually have a higher probability to be routed to cloudlets via a longer path.

The rest is to investigate the impact of the number of cloudlets K on the performance of different algorithms, by varying K from 20 to 100 while fixing n at 200. Figs. 7a and 7b illustrate the curves of the average cloudlet access delays delivered by different algorithms for the problem with and without the identical capacity constraint on the K cloudlets, from which it can be seen that the average cloudlet access delays by algorithms `Heuristic` and `Appro` are much smaller than those by algorithms `Top-`K and `Random`. Also, the average cloudlet access delay by each mentioned algorithm decreases with the increase of K, since each user request will have more chances to be assigned to its nearest cloudlet. The similar performance results of algorithm `Appro-Extension` can be found in Fig. 7c, omitted.

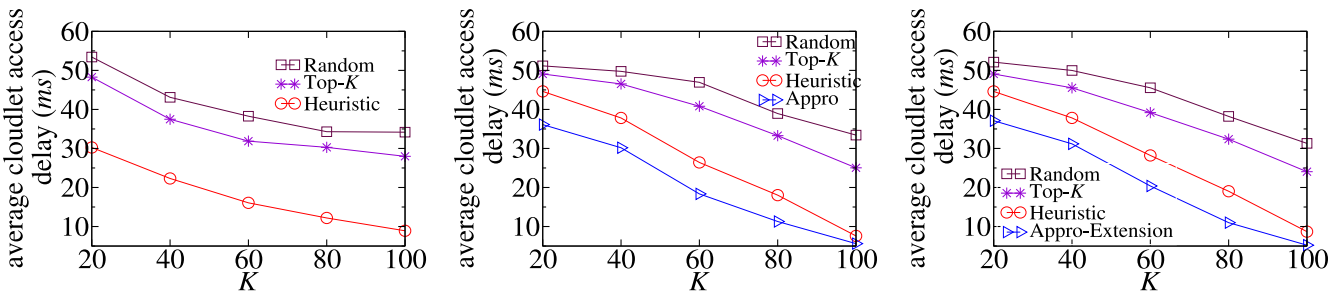
7.4 Performance Evaluation of the Online Algorithm

We finally investigate the performance of algorithm `Online_Assignment` for a monitoring period of 24 hours with each hour as a time slot and fixing the number of APs in the WMAN at 200. We consider two arrival patterns of user requests within the 24-hour monitoring period: (1) the uniform user request pattern, where the number of user requests received at each AP $v \in V$ at each time slot follows the uniform distribution within an interval $[(1-\rho)w(v), (1+\rho)w(v)]$; and (2) the lognormal user request pattern, where the number of user requests arrived at each AP during a day usually starts to rise at around 9:00 am, reaching a peak at around 12:00 pm, and leveling off before 6:00 pm. For simplicity, the uniform user request pattern and lognormal user request pattern are referred to as patterns `Uniform` and `Lognormal` for short.

Fig. 8 plots the curves of average cloudlet access delays by algorithm `Online_Assignment` under two different user request patterns: `Uniform` and `Lognormal`. From Figs. 8 and 4, it can been seen that the average cloudlet access delay by algorithm `Online_Assignment` with `Uniform` request pattern is only no more than twice that by algorithm `Heuristic`. Thus, Fig. 8 indicates that the cloudlet placements delivered by algorithms `Heuristic` and `Appro`, based on the expected number of user requests at each AP is also applicable to the case where the user request fluctuations at APs are insignificant. Also, it can be seen from Fig. 8 that the average cloudlet access delay by algorithm `Online_Assignment` with `Lognormal` has a slight increase starting from 9:00 am to 6:00 pm, due to the user request congestion during that period.

8 Conclusion and Future Works

Cloudlets have been emerged as an important technology that can extend the computing capabilities significantly of resource-constrained mobile devices. In this paper we first studied the capacitated cloudlet placement problem in a large-scale Wireless Metropolitan Area Network with the objective to minimize the average cloudlet access delay between mobile users and the cloudlets serving their requests. We then provided an exact solution to the problem when the problem size is small, otherwise, we proposed a fast yet scalable heuristic for it. For a special case of the problem when all cloudlets have identical computing capacities, we devised two novel approximation algorithms, depending on whether identical resource demands by all user requests. We finally



(a) The impact of K with different cloudlet capacities

(b) The impact of K with identical cloudlet capacities and identical resource demands

(c) The impact of K with identical cloudlet capacities and different resource demands

Fig. 7. The impact of the number of cloudlets K on the performance of different algorithms.

evaluated the performance of the proposed algorithms by experimental simulations. The simulation results showed that the proposed algorithms are very promising. In the future we will study this problem by investigating the delay impact between users and their APs.

Acknowledgments

The authors appreciate the three anonymous referees for their constructive comments and valuable suggestions, which help us significantly improve the quality and presentation of the paper. The work by Song Guo was partially supported by the Strategic International Collaborative Research Program (SICORP) Japanese (JST)—US (NSF) Joint Research "Big Data and Disaster Research" (BDD).

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[3] W. Cai, V. C. M. Leung, and M. Chen, "Next generation mobile cloud gaming," in *Proc. IEEE 7th Int. Symp. Service Oriented Syst. Eng.*, 2013, pp. 551–560.

[4] W. Cai, V. C. M. Leung, and L. Hu, "A cloudlet-assisted multiplayer cloud gaming system," *J. Mobile Netw. Appl.*, vol. 19, pp. 144–152, 2014.

[5] V. Cardellini, V. D. N. Personé, V. D. Valerio, F. Facchinei, V. Grassi, F. L. Presti, and V. Piccial. (2013). A game-theoretic approach to computation offloading in mobile cloud computing. Tech. Rep. [Online]. Available: http://www.optimization-online.org/DB_FILE/2013/08/3981.pdf

[6] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys, "A constant-factor approximation algorithm for the k-median problem," in *Proc. 31st Annu. ACM Symp. Theory Comput.*, 1999, pp. 1–10.

[7] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, 2011, pp. 301–314.

[8] S. Clinch, J. Harkes, A. Friday, N. Davies, and M. Satyanarayanan, "How close is close enough? Understanding the role of cloudlets in supporting display appropriation by mobile users," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2012, pp. 122–127.

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

[10] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl. Services*, 2010, pp. 49–62.

[11] E. Gelenbe, R. Lent, and M. Douratsos, "Choosing a local or remote cloud," in *Proc. 2nd Symp. Netw. Cloud Comput. Appl.*, 2012.

[12] L. Gu, D. Zeng, A. Barnawi, S. Guo, and I. Stojmenovic, "Optimal task placement with QoS constraints in geo-distributed data centers using DVFS," *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 2049-2059, Jul. 1, 2015.

[13] (2000). GT-ITM [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

[14] D. T. Hoang, D. Niyato, and P. Wang, "Optimal admission control policy for mobile cloud computing hotspot with cloudlet," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2012, pp. 3145–3149.

[15] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," To appear in *IEEE Trans. Cloud Comput.*, 2015, Doi: 10.1109/TCC.2015.2449834.

[16] H. Jin, X. Wang, S. Wu, S. Di, and X. Shi, "Towards optimized fine-grained pricing of IaaS cloud platform," *IEEE Trans. Cloud Comput.*, vol. 3, pp. 436–448, Dec. 2015.

[17] N. Karmarkar, "A new polynomial time algorithm for linear programming," *Combinatorica*, vol. 4, pp. 373-395, 1984.

[18] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Proc. 2nd Int. ICST Conf. Mobile Comput., Appl. Services*, 2012, pp. 59–79.

[19] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, 2012, pp. 945–953.

[20] K. Kumar and Y. Lu, "Cloud computing for mobile users: can offloading computation save energy," *J. Comput.*, vol. 43, pp. 51–56, 2010.

[21] LAN/MAN Standards Committee of the IEEE Computer Society, *IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*, IEEE Standard 802-2001, Feb., 2002.

[22] E. A. Lee, et al.,"The swarm at the edge of the cloud," *J. Des. Test*, vol. 31, pp. 8–20, 2014.

[23] S. Li. (2014). An improved approximation algorithm for the hard uniform capacitated k-median problem [Online]. Available: http://arxiv.org/abs/1406.4454

[24] (2013). lp_solve [Online]. Available: http://lpsolve.sourceforge.net/

[25] J. Niu, W. Song, and M. Atiquzzaman, "Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications," *J. Netw. Comput. Appl.*, vol 37, pp. 334–347, 2014.

[26] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *Proc. IEEE INFOCOM*, 2001, pp. 1587–1596.

[27] M.R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proc. 9th Int. Conf. Mobile Syst., Appl. Services*, 2011, pp. 43–56.

[28] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "MuSIC: Mobility-aware optimal service allocation in mobile cloud computing," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, 2013, pp. 75–82.

[29] S. Ren and M. van der Schaar, "Dynamic scheduling and pricing in wireless cloud computing," *IEEE Trans. Mobile Comput.*, vol. 13, no. 10, pp. 2283–2292, Oct. 2014.

[30] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009.

[31] M. Shiraz, S. Abolfazli, Z. Sanaei, and A. Gani, "A study on virtual machine deployment for application outsourcing in mobile cloud computing," *J. Supercomput.*, vol. 63, pp. 946–964, 2013.

[32] T. Verbelen, P. Simoens, F. D. Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proc. 3rd ACM Workshop Mobile Cloud Comput. Services*, 2012, pp. 29–36.

[33] A. Wolbach, J. Harkes, S. Chellappa, and M. Satyanarayanan, "Transient customization of mobile computing infrastructure," in *Proc. 1st Workshop Virtualization Mobile Comput.*, 2008, pp. 37–41.

[34] Q. Xia, W. Liang, and W. Xu, "Throughput maximization for online request admissions in mobile cloudlets," in *Proc. IEEE 38th Conf. Local Comput. Netw.*, 2013, pp. 589–596.

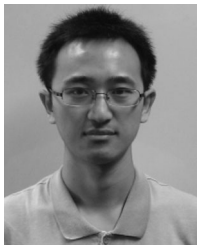
[35] Q. Xia, W. Liang, Z. Xu, and B. Zhou, "Online algorithms for location-aware task offloading in two-tiered mobile cloud environments," in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput.*, 2014, pp. 109–116.

[36] Z. Xu and W. Liang, "Operational cost minimization for distributed data centers through exploring electricity price diversity," *Comput. Netw.*, vol. 83, pp. 59–75, 2015.

[37] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Capacitated cloudlet placements in wireless metropolitan area networks," in *Proc. 40th Annu. IEEE Conf. Local Comput. Netw.*, 2015.

[38] H. Yin, X. Zhang, T. Zhan, Y. Zhang, G. Min, and D. Wu, "NetClust: A framework for scalable and pareto-optimal media server placement," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 2114–2124, Dec. 2013.

[39] Y. Zhang, H. Liu, L. Jiao, and X. Fu, "To offload or not to offload: An efficient code partition algorithm for mobile cloud computing," in *Proc. IEEE 1st Int. Conf. Cloud Netw.*, 2012, pp. 80–86.

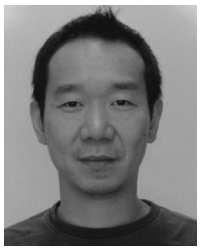


Zichuan Xu received the BSc and ME degrees from Dalian University of Technology in China in 2008 and 2011, both in computer science. He is currently working toward the PhD degree in the Research School of Computer Science, Australian National University. His research interests include distributed clouds, data center networks, cloud computing, algorithmic game theory, and optimization problems. He is a student member of the IEEE.

Weifa Liang (M'99-SM'01) received the BSc degree from Wuhan University, China in 1984, the ME degree from the University of Science and Technology of China in 1989, and the PhD degree from the Australian National University in 1998, all in computer science. He is currently a full professor in the Research School of Computer Science, Australian National University. His research interests include design and analysis of routing protocols for wireless ad hoc and sensor networks, cloud computing, graph databases, design and analysis of parallel and distributed algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

Wenzheng Xu received the BSc, ME, and PhD degrees from Sun Yat-Sen University, Guangzhou, China, in 2008, 2010, and 2015, respectively, all in computer science. He currently is a special associate professor at the Sichuan University, and was a visitor at the Australian National University. His research interests include wireless sensor networks, approximation algorithms, graph theory, online social networks, cloud computing, and mobile computing. He is a member of the IEEE.

Mike Jia received the BSc degree in mathematics and computer science from Imperial College London, United Kingdom, in 2013, and the honors in computer science from the Australian National University in 2014. He is a first year PhD student in computer science at the Australian National University. His research interests include mobile cloud computing and software-defined networks. He was briefly a research assistant in the Department of Computing, Hong Kong Polytechnic University in 2013.



Song Guo (M'02-SM'11) received the PhD degree in computer science from the University of Ottawa, Canada, in 2005. He is currently a full professor at the School of Computer Science and Engineering, University of Aizu, Japan. His research interests are mainly in the areas of protocol design and performance analysis for wireless networks and distributed systems. He has published over 250 papers in refereed journals and conferences in these areas and received three IEEE/ACM Best Paper Awards. He currently serves as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*, an associate editor of the *IEEE Transactions on Emerging Topics in Computing* for the track of Computational Networks, and on editorial boards of many others. He has also been in organizing and technical committees of numerous international conferences. He is a senior member of the IEEE and the ACM.