This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2907605, IEEE Internet of Things Journal

Cloudlet Placement and Task Allocation in Mobile Edge Computing

Song Yang, *Member, IEEE,* Fan Li, *Member, IEEE,* Meng Shen, *Member, IEEE,* Xu Chen, *Member, IEEE,* Xiaoming Fu, *Senior Member, IEEE,* Yu Wang, *Fellow, IEEE*

***Abstract*—Mobile Edge Computing (MEC) offers a way to shorten the cloud servicing delay by building the small-scale cloud infrastructures such as cloudlets at the network edge, which are in close proximity to end users. On one hand, it is energy consuming and costly to place each cloudlet on each Access Point (AP) to process the requested tasks. On the other hand, the service provider should provide delay-guaranteed service to end users, otherwise they may get revenue loss. In this paper, we first model how to calculate the task completion delay in MEC and mathematically analyze the energy consumption of different equipments in MEC. Subsequently, we study how to place cloudlets on the network and allocate each requested task to cloudlets and public cloud with the minimum total energy consumption without violating each task's delay requirement. We prove that this problem is NP-hard and propose a Benders Decomposition-based algorithm to solve it. We also present a Software-Defined Network (SDN)-based framework to deploy the proposed algorithm. Extensive simulations reveal that the proposed algorithm can achieve an (close-to-)optimal performance in terms of energy consumption and acceptance ratio compared with two benchmark heuristics.**

***Index Terms*—Mobile edge computing, cloudlet placement, task allocation, delay, energy consumption.**

I. Introduction

With the development of mobile computing technology, more and more newly mobile applications (e.g., argument reality and online gaming) have been emerging. However, the mobile devices usually suffer from battery and memory limitations and need to offload the tasks on remote located datacenter, which is known as cloud computing [1]. Although cloud computing can provide a virtually unlimited capability service to end users, the requested task from end users may experience long-distance and congested transmission to the remote public cloud and incur higher servicing delay. Cloud computing hence cannot always satisfy the ever-increasing stringent delay task demands [2], which remains a crucial drawback to tackle.

The concept of Mobile Edge Computing (MEC) [3] (or Fog Computing[1]) has been proposed to bring the computing resources closer to end users by installing small resource-limited cloud infrastructures such as cloudlets at the network edge. In this context, the end users' task workloads can be offloaded on cloudlets (which can be called edge layer) by achieving short service delay. Considering that the capacity of the cloudlet is limited, the cloudlets are usually connected via e.g., cellular core network [5] in order to expose services to more nearby end users by efficiently utilizing the capacity and load of cloudlets. Moreover, when the edge layer's task processing capacity is full, the edge layer also needs to be connected with remote public cloud (which can be called cloud layer) to further offload the task workloads. The whole MEC architecture can be see in Fig. 1. MEC therefore embraces characteristics like location awareness, real-time task processing, agile network management and control.

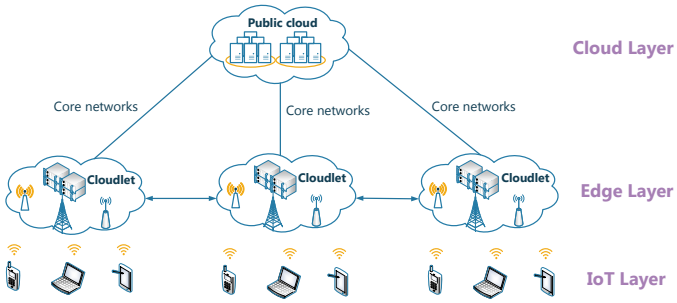


Fig. 1: A Mobile Edge Cloud framework.

On one hand, considering that there are numerous Access Points (AP) spots, it is impossible and not necessary to place the cloudlet on each AP spot due to the economic cost and energy concerns such as electricity bills and environmental pollution. On the other hand, it is also important and desirable to guarantee each task's delay requirement. Especially with the maturation of mobile technology and advancement of Internet of Things (IoT) technology, more and more end users usually ask for delay-stringent tasks [6] or applications [7] such as online gaming and data processing. In this sense, the service vendor may get revenue loss or even lose clients [8] if the

S. Yang, F. Li, and M. Shen are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China (email: {S.Yang, fli, shenmeng}@bit.edu.cn).

X. Chen is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China (email: chenxu35@mail.sysu.edu.cn).

X. Fu is with Institute of Computer Science, University of Göttingen, Göttingen, Germany (email: Fu@cs.uni-goettingen.de)

Y. Wang is with Department of Computer Science, University of North Carolina at Charlotte, NC (email: yu.wang@uncc.edu)

[1]According to [4], "the key difference between the edge computing and fog computing is exactly where that intelligence and computing power is placed. Fog computing pushes intelligence down to the local area network level of network architecture, processing data in a fog node or IoT gateway. Edge computing pushes the intelligence, processing power and communication capabilities of an edge gateway or appliance directly into devices like programmable automation controllers (PACs)."

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2907605, IEEE Internet of Things Journal

task delay requirement is not satisfied. With the exponentially increasing trend of IoT applications and tasks requested from end users, more equipments and devices in edge layer and cloud layer are needed to be switched on to process these requests, which accordingly leads to considerable energy consumption. As a result, it is important for us to explore the trade-off between energy consumption and service delay in MEC by strategically placing cloudlets and scheduling tasks among cloudlets and public cloud. In this paper, we study how to place cloudlets on APs and schedule users' tasks with the minimum energy consumption such that each task' delay requirement is satisfied. Our key contributions are as follows:

- We mathematically analyze the task completion time and the energy consumption of different equipments in MEC.
- We define the cloudlet placement and task allocation problem that takes energy and delay into account and show it is NP-hard.
- We propose an exact Mixed Integer Linear Programming (MILP) formulation and a Benders Decomposition-based solution to solve the proposed problem. We accordingly propose a Software-Defined Network (SDN)-based framework to deploy the proposed algorithm.
- We conduct simulations to validate the performances of the proposed algorithm with two benchmark heuristics.

The remainder of this paper is organized as follows. Section II presents the related work. Section III mathematically models the task completion time in MEC and analyzes energy consumption of different equipments in MEC. Section IV defines the Cloudlet Placement and Task Allocation (CPTA) problem, mathematically formulates the CPTA problem as an Mixed Integer Linear Programming (MILP) and shows that it is NP-hard. In Section V, we propose a Benders decomposition method to solve the CPTA problem and A SDN-based algorithm deployment framework. Section VI provides the simulation results and we conclude in Section VII.

II. Related Work

A comprehensive and detailed survey about MEC can be found in [9], [10], [11]. Moreover, [12], [13] and the compared survey papers therein provide overviews about fog computing.

Jia *et al.* [14] study how to place K cloudlets and allocate user to cloudlets in Wireless Metropolitan Area Networks (WMAN) such that the average system response time is minimized. In [14], the cloudlets are modeled as a M/M/c queue and the requested tasks of users arrive according to the Poisson process. They [14] further propose a Density-Based Clustering (DBC) algorithm to solve this problem. Under the same delay model with [14] and assuming that the cloudlets are already deployed on networks, Jia *et al.* [15] propose a fast heuristic and a distributed genetic algorithm to schedule workload such that the maximum task response time is minimized. Ma *et al.* [16] model the whole MEC as a queuing network, and formulate the workload scheduling problem in MEC as a convex problem. They further present an efficient algorithm by exploiting the linear property of constraints in the convex formulation. In [17], it is defined that the average cloudlet access delay as the whole average service time (a M/M/1 queue for each AP is assumed), and the cloudlet processing time as well as flow traversing time are not considered, which is not very practical. Zhao *et al.* [17] propose a ranking-based heuristic to deploy K cloudlets to minimize the average cloudlet access delay. The above papers adopt a queueing-based model to reflect and calculate the whole system delay instead of caring about each individual task request's delay requirement. Moreover, according to the queueing theory, it is required in the above papers that the cloudlet processing delay follows an exponential distribution and the arrival of task requests follow a Poisson distribution.

Ma *et al.* [18] study how to minimize the number of placed cloudlets in order that all the task requests can be allocated and provisioned. However, only the fixed cloud access delay is considered in [18]. Meng *et al.* [19] tackle the cloudlet placement and request scheduling problem, which is formulated as a problem variant of K-facility location problem. They first propose an approximation algorithm to solve this problem, and then devise an online algorithm to deal with how to schedule requests when the cloudlets are placed. However, Meng *et al.* [19] only consider the task transmitting delay as the whole task completion delay. Wang *et al.* [20] study how to place K edge servers by proposing an exact Integer Linear Programming (ILP) solution in order to minimize the total access delay, which is proportional to the distance between the base station and edge server. Xu *et al.* [21] propose an approximation algorithm to solve the K cloudlet placement problem in WMAN such that the average access delay from AP to cloudlet is minimized. In [22], it is assumed that when a cloudlet is deployed at a BS, a certain cost will be consumed. Fan and Ansari [22] study the cloudlet placement and cloudlet servers installation problem to jointly minimize the costs and the total end-to-end transmitting delay by proposing an ILP. The above papers do not model task completion delay thoroughly. For example, the task processing delay in cloudlet and in the public cloud (when the tasks are offloaded to the public cloud) is not considered. In this paper, we propose a general model to calculate the task completion delay by considering both task transmitting delay and task processing delay under a further mild assumption that each task is divisible.

The most relevant work with us are [5], [23], [24]. Sun and Ansari [5] address the problem of Avatar placement in cloudlets such that the energy consumption is minimized and the end-to-end delay for each user's request is satisfied. The avatar means a private virtual machine executing in the cloudlet for offloading user's tasks. They devise an efficient Green-aware Avatar Placement heuristic to solve the problem. Xiao and Krunz [23] propose a distributed Alternating Direction Method of Multipliers (ADMM)-based algorithm to solve the workload/task allocation problem such that the delay (which is modelled according to the queueing theory) is satisfied under the given power efficiency. However, both [5] and [23] do not consider the scenario of offloading tasks from cloudlet to public cloud. Deng *et al.* [24] first formulate the delay-aware workload allocation problem in fog-cloud networks with the minimum total power consumption as a non-convex problem. They further decompose this problem into 3

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2907605, IEEE Internet of Things Journal

subproblems that can be efficiently solved in 3 subdomains, namely (1) edge layer, (2) cloud layer and (3) communication path domain between edge layer and cloud layer. They respectively apply Interior-Point methods, Bender decomposition and Hungarian method to solve these 3 subproblems. However, [24] does not consider the cloudlet placement problem. Moreover, due to the fact that they [24] apply queueing theory to model the task completion delay, the overall system delay is to be less than a specified value instead of satisfying each task request's delay requirement, as we do in this paper.

III. System Model

A. Task Completion Delay Calculation

Assume that there is a set of N inter-connected Access Points (AP) $\mathcal{N}$, and for each AP pair $i, j \in \mathcal{N}$, the wired link (i, j) connecting them[2] provides a fixed transmission rate of $\eta(i, j)$ for each task flow. For each AP $n \in \mathcal{N}$, a set of users sends his/her task to n via the wireless access network. For simplicity, we assume that the task sent from one user can only be received by one associated AP. Moreover, we assume that each end user can transmit the task to the AP via an unique available channel. We therefore do not take the wireless delay from users to AP into account for brevity in this paper, since this part of value only depends on the channel bandwidth and task size [25] in this context and hence cannot be further optimized/minimized. In order to provision the requested tasks, we need to place a certain number of cloudlets on AP spots and allocate the tasks on the APs located with cloudlets. We denote by $\mathcal{C}$ the set of public cloud and we assume there is only one public cloud without loss of generality. For ease of expression, we denote $\mathcal{V} = \mathcal{N} \bigcup \mathcal{C}$ by the set of total $V = N + 1$ network nodes. Table I gives all the notations used in this paper.

TABLE I: Notations.

Notation	Description
$\mathcal{V}, \mathcal{N}, \mathcal{C}$	The set of total network nodes, AP nodes, public cloud
(i, j)	The link connecting node i and j
$\eta(i, j)$	The transmission rate over link (i, j)
$\alpha(n)$	Computation capability of AP n with cloudlet placed on it
R	The set of task requests. For each $r(\phi, \Delta, f) \in R$, ϕ indicates the AP that request r directly accesses, Δ represents the delay requirement, f denotes the task size
P_o, P_t	Idle power consumption, workload-dependent power factor
T_s	Device switched-on duration time
X_n^r	A float variable indicating the amount of allocated tasks for r on the AP node n
Y_n^r	A float variable meaning the amount of allocated tasks for r relaying from AP node n to the public cloud $\mathcal{C}$
Z_n	A boolean variable indicating whether a cloudlet is placed on AP n
W_{ij}	A boolean variable indicating whether link (i, j) is in use

If a cloudlet is placed on AP n, then we assume that n has a computation capability (a.k.a., available computing slots [26]) of $\alpha(n)$. If the task workload on n exceeds $\alpha(n)$, n can (i) offload the remaining tasks to the other AP nodes with cloudlets located on them, and/or (ii) offload the remaining tasks to the public cloud via a unique Wide Area Network (WAN) path[3]. A task request r is represented by $r(\phi, \Delta, f)$, where ϕ denotes its connected accessing AP, Δ indicates the required task completion delay and f means the task size[4]. A request task r is assumed to be roughly "infinitesimally divisible"[5] [27], [28], [29], [30], which means that different fractions of r can be performed at multiple cloudlets. For example, in data partitionable tasks [31], a simple-to-describe operation (e.g., read or write) can independently process/access different blocks of data.

As a result, the delay [26] for transmitting (a portion of) task with size of f on link (i, j) is calculated as follows:

$$d_{i,j}(f) = \frac{f}{\eta(i, j)} \tag{1}$$

where $\eta(i, j)$ denotes the transmission rate over link (i, j).

Formally, if $r(\phi, \Delta, f)$ is processed by m cloudlets u_1, u_2, …, u_m deployed on nodes n_1, n_2, …, n_m to execute tasks f_1, f_2, …, f_m on these APs and relay remaining tasks f'_1, f'_2, …, f'_m to the cloud such that $\sum_{x=1}^{m}(f_x + f'_x) = f$, then the delay for provisioning the requested task $r(\phi, \Delta, f)$ can be calculated as follows:

$$D = \max_{x=1}^{m} D_x \tag{2}$$

where

$$D_x = \begin{cases} d_{\phi,n_x}(f_x) + g(f_x) & \text{if } f'_x = 0 \\ d_{\phi,n_x}(f_x + f'_x) + g(f_x) + d_{n_x,\mathcal{C}}(f'_x) + h(f'_x) & \text{otherwise} \end{cases} \tag{3}$$

where $g(f_x)$ and $h(f'_x)$ denote the processing time on cloudlet for workload f_x and on public cloud $\mathcal{C}$ for workload f'_x, respectively. For a request r, we let f_x represent the amount of task executed on cloudlet node x, and f'_x indicate the amount of task relaying from cloudlet node n_x to the public cloud $\mathcal{C}$. In this sense, $f'_x > 0$ implies that cloudlet node x has reached its maximum computation capability limit and has to relay f'_x amount of task to the public cloud $\mathcal{C}$. We assume that $g(f)$ and $h(f)$ have a linear relationship with f. For simplicity, we set $g(f) = \theta_u \cdot f$ and $h(f) = \theta_v \cdot f$, where θ_u and θ_v denote the coefficients[6] that reflect the equipment's computation capability (e.g., CPU cycles per second) [32].

To better illustrate it, we take Fig. 2 for an example, where we set $\theta_u = 0.1$ for all the cloudlets and $\theta_v = 0.05$. Assuming that AP_1 receives a task with the size of 50. Since there is no cloudlet on AP_1, it has to relay this request to the AP nodes with cloudlet located on them. In this example, either AP_2 or AP_3 cannot process the whole task because of their limited computation capability, therefore we split r into two parts and send tasks with size of $f_1 = 40$ and $f_2 = 10$

[2]In practice, (i, j) may represent a path traversing several links.

[3]Due to cost concerns, we assume that only the AP node with cloudlet is connected to the public cloud with a WAN path.

[4]For simplicity, here the task can represent the input data that needs to be processed by cloudlet or public cloud and hence f is in the unit of Gb.

[5]For a task that is divisible, we assume that each individual divisible basic unit of this task is sufficiently small in this paper. In this context, although we apply two float variables X_n^r and Y_n^r in the following proposed ILP and the Benders decomposition approach, we can roughly round the returned solutions to a certain decimal places by losing a little bit precision/accuracy.

[6]In this paper, we assume that the cloudlet and public cloud allocate the same computing resource to each task and therefore $g(f)$ and $h(f)$ do not change.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2907605, IEEE Internet of Things Journal

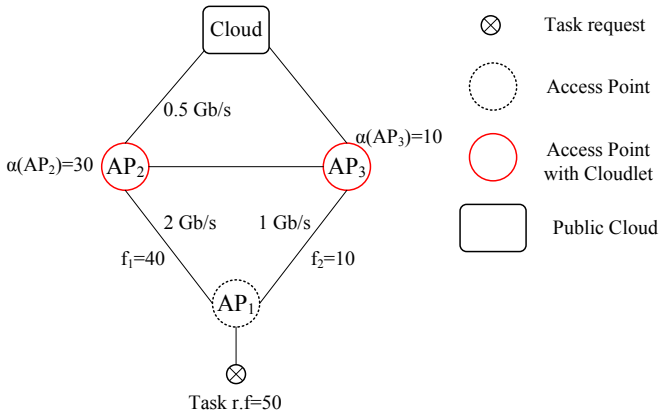


Fig. 2: An example of calculating task completion delay.

to AP_2 and AP_3, respectively. Furthermore, considering that $\alpha(AP_2) < 40$ it has to relay the remaining task to the public cloud for further processing. The task completion time is $\underbrace{\frac{40}{2}+\frac{10}{0.5}}_{\text{transmitting delay}} + \underbrace{30*0.1++10*0.05}_{\text{Processing delay}} = 43.5$ for f_1. Similarly, the task completion time for f_2 is $\frac{10}{1}+10*0.1 = 11$. Consequently, the task completion delay is $\max(11, 43.5) = 43.5$.

B. Energy Consumption

Often, e.g., in [5], [33], [34], it is assumed that power consumption P calculation for a network equipment consists of two parts and has the following linear relationship with the traffic/task workload:

$$P = P_o + P_t \cdot \omega \tag{4}$$

where P_o is the overhead which represents idle power consumption of a switched-on device in the unit of Watt, P_t represents the traffic/task workload-dependent power factor in the unit of Watt/Gbps, and ω denotes the amount of traffic/task workload in the unit of Gbps. For example, a server's power consumption is calculated as [35]:

$$P_{sr}(b) = P_o^{sr} + \frac{P_m^{sr} - P_o^{sr}}{C_{sr}} \cdot \omega \tag{5}$$

where P_o^{sr} and P_m^{sr} represent the power consumption of the server when it is idle and fully loaded, respectively, C_{sr} denotes its maximum processing capacity and ω stands for its current load ($\omega \leq C_{sr}$). Denote $P_t^{sr} = \frac{P_m^{sr} - P_o^{sr}}{C_{sr}}$ as the traffic/task workload dependent power factor, according to [36], $C_{sr} = 1.8$ Gbps, $P_o^{sr} = 325\ W$ and $P_m^{sr} = 380\ W$. Therefore, the power consumption of a server is:

$$P_{sr} = 325 + 30.6 \cdot \omega \tag{6}$$

where $0 \leq \omega \leq 1.8$.

In this paper, we use E to represent the energy consumption in the unit of Wh (or kWh), and P to indicate the power consumption in the unit of W (or kW). Consequently, the energy consumption can be expressed as $E = P_o \cdot T_s + P_t \cdot \omega \cdot T_d$, where T_s and T_d symbolize the device switching-on and task processing time in the units of hours[7] ($T_s >= T_d$), respectively. As a result, the energy consumption can be calculated as:

$$E = P_o \cdot T_s + P_t \cdot \omega \cdot T_d = P_o \cdot T_s + P_t \cdot f \tag{7}$$

where f indicates the task size in the unit of Gb and E is in the unit of Wh. Accordingly, $P_o \cdot T_s$ indicates the idle energy consumption of a device for switching on T_s hours, and $P_t \cdot \omega \cdot T_d$ represents the energy consumption for processing the task with size of $\omega \cdot T_d = f$ Gb. In particular, P_t is in the unit of Watt/Gbps and f is in the unit of Gb. In this sense, $P_t \cdot f$ means the task processing-dependent energy consumption with the unit of Wh (we transfer the unit of seconds to the unit of hours), and $P_o \cdot T_s$ indicates the idle energy consumption. $P_o \cdot T_s + P_t \cdot f$ yields the total energy consumption. From Eq. (7) we see that the larger the task size, the more energy consumption for cloudlet/cloud to process it.

In this paper, we consider 3 main energy-consuming equipments, namely (1) links (ij), (2) public cloud and (3) cloudlets (clt). More specifically, the energy consumption of the link (i, j) which connects two APs (or AP with located cloudlet and public cloud) is traffic independent. That is, as long as link (i, j) is switched on for transmitting data it consumes constant energy [37], i.e., $P_t^{ij} = 0$ for the link. The public cloud consists of a cluster of servers which are organized in a tree-like topology. The servers are always turned on in case of dealing with traffic fluctuation which means that the idle energy consumption of servers is constant and cannot be minimized/saved. In this sense, we discard the contribution of P_o^{cloud} and regard $P_o^{\text{cloud}} = 0$ in this paper. On the other hand, we have $P_o^{clt} > 0$ and $P_t^{clt} > 0$ for the cloudlet.

IV. Problem Definition and Complexity Analysis

A. Problem Definition and Formulation

Formally, the Cloudlet Placement and Task Allocation (CPTA) problem can be defined as follows:

Definition 1: Given are a set of APs $\mathcal{N}$, a public cloud $\mathcal{C}$, and a set of requests R. The Cloudlet Placement and Task Allocation (CPTA) problem is to place cloudlets on $\mathcal{N}$ and allocate each user's task to corresponding cloudlets and public cloud such that the total energy consumption is minimized and each task's delay requirement is satisfied.

The considered CPTA problem in this paper actually happens in the network planning stage. In this paper, we assume that all the task requests R are statistically collected from a whole service period (e.g., 1 day or 1 month) and processed for instance by taking the average at a reasonable service time slot over a whole service period. As a result, R can reflect the average task request pattern in the network and our aim in this paper is to place cloudlets on the network with minimum total energy consumption such that all these task requests are

[7]When the device switching-on time or task processing time is less than one hour, for example, 30 minutes, we can convert it to 0.5 hour for the ease of calculating the energy consumption in the unit of Wh (or kWh).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2907605, IEEE Internet of Things Journal

satisfied regarding their QoS requirement[8] (task completion delay). When the cloudlets are placed in network, the task allocation problem happens in short-term continuously. But since the cloudlet placement design was based on average historical task requests, it can efficiently serve the task requests which arrive dynamically by guaranteeing the required task completion delay requirement.

We first formulate the CPTA problem as an exact Mixed Integer Linear Programming (MILP). We begin with some necessary notations and variables.

R: A set of task requests.

X_n^r: The amount of allocated tasks for r on the AP node n.

Y_n^r: The amount of allocated tasks for r relaying from AP node n to the public cloud $\mathcal{C}$.

Z_n: A boolean variable and it is 1 (true) if a cloudlet is placed on AP n, and 0 otherwise.

W_{ij}: A boolean variable and it is 1 (true) if link (i,j) is in use, and 0 otherwise.

Objective:

$$\min \quad \sum_{n\in\mathcal{N}} Z_n \cdot P_o^{clt} \cdot T_s + \sum_{r\in R, n\in\mathcal{N}} X_n^r \cdot P_t^{clt} + \sum_{i,j\in\mathcal{N}:i<j} W_{ij} \cdot P_o^{ij} \cdot T_s + \sum_{r\in R, n\in\mathcal{N}} Y_n^r \cdot P_t^{Cloud} \quad (8)$$

Delay Constraint:

$$\frac{X_n^r + Y_n^r}{\eta(r.\phi, n)} + \theta_u \cdot X_n^r + \frac{Y_n^r}{\eta(n,\mathcal{C})} + \theta_v \cdot Y_n^r \le r.\Delta \quad \forall r \in R, n \in \mathcal{N} : n \neq r.\phi \quad (9)$$

Task Constraint:

$$\sum_{n\in\mathcal{N}} (X_n^r + Y_n^r) == r.f \quad \forall r \in R \quad (10)$$

Cloudlet Capacity Constraint:

$$\sum_{r\in R} X_n^r \le \alpha(n) \quad n \in \mathcal{N} \quad (11)$$

Task Allocation Constraints:

$$M \cdot Z_n \ge \sum_{r\in R} (X_n^r + Y_n^r) \qquad \forall n \in \mathcal{N} \quad (12)$$

$$M \cdot W_{ij} \ge \sum_{r\in R: r.\phi = i} (X_j^r + Y_j^r) + \sum_{r\in R: r.\phi = j} (X_i^r + Y_i^r) \quad \forall i,j \in \mathcal{N} : i \neq j \quad (13)$$

Eq. (8) minimizes the overall energy consumption. More specifically, $\sum_{n\in\mathcal{N}} Z_n \cdot P_o^{clt} \cdot T_s + \sum_{r\in R, n\in\mathcal{N}} X_n^r \cdot P_t^{clt}$ means the energy consumption of all the placed cloudlets, $\sum_{i,j\in\mathcal{N}:i<j} W_{ij} \cdot P_o^{ij} \cdot T_s$ denotes the energy consumption of all the active links, and $\sum_{r\in R} Y_n^r \cdot P_t^{Cloud}$ represents the energy consumption of cloud servers. Eq. (9) ensures that each task's delay requirement is not violated. In order to make this constraint linear, instead of using max function, we let the delay of task which is executed on each node be less than the specified value, which is equivalent to Eq. (2). Moreover, we set $\eta(n,n)$ be an infinite large number. Therefore, for a request r, if a cloudlet is placed on $r.\phi$, then the transmitting delay can be neglected. Eq. (10) ensures that the total allocated tasks on cloudlets and public cloud for each request is equal to its required task size. Eq. (11) ensures that the total allocated tasks on one cloudlet do not exceed its maximum computation capacity. Eq. (12) identifies whether an AP is placed by a cloudlet by applying big-M method. By setting M to be a large enough number, Z_n is equal to be 1 if there is positive amount of task on n or relayed from n to the cloud, and 0 otherwise. More specifically, the maximum value of $\sum_{r\in R}(X_n^r + Y_n^r)$ is $2|R|$. Since M is a sufficiently large number, it holds that $M > 2|R|$. In this sense, when $X_n^r = 1$ and/or $Y_n^r = 1$, we impose $Z_n = 1$ in order to make Eq. (12) to hold. Otherwise, when $X_n^r = 0$ and $Y_n^r = 0$, either $Z_n = 0$ or $Z_n = 1$ will make Eq. (12) to be true. But $Z_n = 0$ in this context, since one term ($\sum_{n\in\mathcal{N}} Z_n \cdot P_o^{clt} \cdot T_s$) of objective function needs to be minimized. Similarly, Eq. (13) identifies whether link (i,j) is switched on by applying big-M method.

B. Complexity Analysis

Theorem 1: The CPTA problem is NP-hard.

Proof 1: Let us first introduce the Capacitated Facility Location (CFL) problem: Given are a set of potential facility sites $\mathcal{N}$ where a facility can be opened with a limited capacity, and a set of requests that must be serviced. The request can be provisioned by multiple facilities, with each facility serving a portion of the request. The CFL problem is to pick a subset of facilities to open in order to minimize the sum of distances from each request to its facility and the sum of opening costs of the facilities. The CFL problem is proved to be NP-hard [38].

We assume that the task delay requirement is satisfied as long as the task is executed on cloudlets, and it will be violated if the task is executed on the public cloud. Now if we correspond the energy consumption of the cloudlets (where P_t^{clt} is set to be a negligible tiny value) and links in the CPTA problem to the facility opening costs and distances from the request to its serving facility in the CFL problem, then CPTA can be reduced to the CFL problem, which is also NP-hard. The proof is therefore complete.

We mention that the considered CPTA problem is different from and more difficult than the CFL problem. More specifically, in the CPTA problem it is required that the task completion delay consisting of transmission delay and processing delay for each request should be satisfied. But this constraint is not imposed in the CFL problem. As a result, the existing approximation algorithms cannot be used to solve the considered CPTA problem with proved bounded approximation ratio. According to [38], if a problem is strongly NP-hard, then it indicates that this problem does not admit the

[8]In this paper, all the task requests are treated/accommodated equivalently (with the same allocated computing resource from cloudlet/cloud) and are not further differentiated.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2907605, IEEE Internet of Things Journal

Fully Polynomial Time Approximation Algorithm (FPTAS)[9]. Since the CFL problem does not admit FPTAS [38], and CPTA problem is more difficult than the CFL problem, the CPTA problem is therefore strongly NP-hard. Furthermore, the MILP formulation in Eqs. (8)-(13) for the CPTA problem includes four variables (2 integer variables and 2 continuous variables) and associated constraints. The scale and complexity of the MILP formulation reveal that the CPTA problem is very difficult to solve and the approximation algorithm can hardly exist to solve it.

V. Solution

In this section, we first present a Benders decomposition approach to solve the CPTA problem. Subsequently, we propose a SDN-based framework to deploy the proposed algorithm.

A. A Benders Decomposition Approach

The mathematical formulation in Eqs.(8)-(13) is a MILP, since it consists of both continuous variables (X, Y) and integer variables (Z, W). When the problem size increases, its running time is exponential which is not efficient. Therefore we apply Benders decomposition approach [39] to Eqs.(8)-(13) to derive a more efficient solution. The Benders decomposition approach partitions the variables of the original problem into two subsets. The first stage master integer linear problem is solved over the first set of variables, and the second stage linear programming subproblem is solved based on the second set of variables and output of the first stage problem. Accordingly, the subproblem provides feasible cut constraint (if the linear programming is bounded) or infeasible cut constraint (if the linear programming is unbounded) to the master problem. Obviously, the master problem yields a Lower Bound (LB), while the subproblem produces an Upper Bound (UB). These two problems are solved iteratively until $UB - LB$ approaches zero (or a given gap ϵ). An illustration of Benders decomposition principle [40] is shown in Fig. 3.

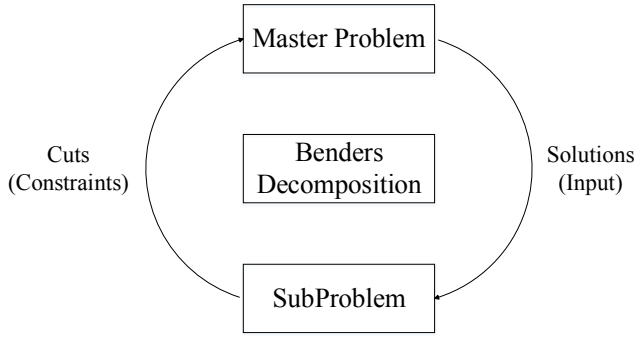


Fig. 3: A Benders decomposition principle illustration.

In our scenario, we first derive the subproblem as Eq. (14) shows. After, we take the dual of Eq. (14) and it is seen in Eq. (15). For clearness, we use $U(Z, W, \vec{\lambda})$ to denote the objective value of the dual.

$$\min \sum_{r\in R, n\in\mathcal{N}} X_n^r \cdot P_t^{clt} + \sum_{r\in R} Y_n^r \cdot P_t^{Cloud}$$
$$s.t. \begin{cases} \frac{X_n^r+Y_n^r}{\eta(r.\phi,n)} + \theta_u \cdot X_n^r + \frac{Y_n^r}{\eta(n,\mathcal{C})} + \theta_v \cdot Y_n^r \le r.\Delta \\ \forall r \in R, n \in \mathcal{N} \\ \sum_{n\in\mathcal{N}} (X_n^r + Y_n^r) == r.f \quad \forall r \in R \\ \sum_{r\in R} X_n^r \le \alpha(n) \quad \forall n \in \mathcal{N} \\ M \cdot \hat{Z}_n \ge \sum_{r\in R} (X_n^r + Y_n^r) \qquad \forall n \in \mathcal{N} \\ M \cdot \hat{W}_{ij} \ge \sum_{r\in R: r.\phi=i} (X_j^r + Y_j^r) + \sum_{r\in R: r.\phi=j} (X_i^r + Y_i^r) \\ \forall i, j \in \mathcal{N} : i \ne j \end{cases} \tag{14}$$

$$\max \quad U(Z, W, \vec{\lambda}) = \sum_{r\in R, n\in\mathcal{N}} r.\Delta \cdot \lambda_{r,n}^1 - \sum_{r\in R} r.f \cdot \lambda_r^2 + \sum_{n\in\mathcal{N}} \alpha(n) \cdot \lambda_n^3 + \sum_{n\in\mathcal{N}} M \cdot \hat{Z}_n \cdot \lambda_n^4 + \sum_{i,j\in\mathcal{N}} M \cdot \hat{W}_{ij} \cdot \lambda_{ij}^5$$
$$s.t. \begin{cases} \sum_{r\in R, n\in\mathcal{N}} (\frac{1}{\eta(r.\phi,n)} + \theta_u) \cdot \lambda_{r,n}^1 - \sum_{r\in R} \lambda_r^2 + \\ \sum_{n\in\mathcal{N}} \lambda_n^3 + \sum_{n\in\mathcal{N}} \lambda_n^4 + \sum_{i,j\in\mathcal{N}} \lambda_{ij}^5 \ge P_t^{clt} \\ \sum_{r\in R, n\in\mathcal{N}} (\frac{1}{\eta(r.\phi,n)} + \frac{1}{\eta(n,\mathcal{C})} + \theta_v) \cdot \lambda_{r,n}^1 - \sum_{r\in R} \lambda_r^2 + \\ \sum_{n\in\mathcal{N}} \lambda_n^4 + \sum_{i,j\in\mathcal{N}, r\in R: r.\phi=i \| r.\phi=j} \lambda_{ij}^5 \ge P_t^{Cloud} \\ \lambda_{r,n}^1, \lambda_r^2, \lambda_n^3, \lambda_n^4, \lambda_{ij}^5 \ge 0, \quad \forall i, j, n \in \mathcal{N}, r \in R \end{cases} \tag{15}$$

After that, Eq. (16) denotes the master problem in the Benders decomposition. More specifically, we let π represent the maximum value of the subproblem, and H be the objective value of the master problem to be minimized. In particular, $U(Z, W, \vec{\lambda}_p) \le \pi$ represents the cut of constraint when Eq. (14) returns a feasible solution, and $U(Z, W, \vec{\lambda}_y) \le 0$ adds the cut of constraint when Eq. (14) yields an unbounded solution.

$$\min \ H = \sum_{n\in\mathcal{N}} Z_n \cdot P_o^{clt} + \sum_{i,j\in\mathcal{N}: i<j} W_{ij} \cdot P_o^{ij} + \pi$$
$$s.t. \begin{cases} U(Z, W, \vec{\lambda}_p) \le \pi \quad \forall \vec{\lambda}_p \in \mathcal{C}_p \\ U(Z, W, \vec{\lambda}_y) \le 0 \quad \forall \vec{\lambda}_y \in \mathcal{C}_y \\ Z_n, W_{ij} \in \{0, 1\} \quad \forall n, i, j \in \mathcal{N} \end{cases} \tag{16}$$

As a result, we propose a Benders Decomposition-based cloudlet Placement and task Allocation (BDPA) algorithm to solve the CPTA problem. More specifically in Algorithm 1, we initialize UB and LB which represent the upper bound and lower bound of the original problem. $\mathcal{C}_p$ and $\mathcal{C}_y$ denote extreme point and extreme ray cut set and will be added as the constraints to the master problem. As long as $UB - LB$ is bigger than ϵ, Step 2-11 is going to find a solution based on Benders decomposition iteratively. In each iteration k, BDPA solves the master problem to obtain $Z^{(k)}$, $W^{(k)}$ and $\pi^{(k)}$ in Step 3 and assigns the objective value H^k to LB in Step 4.

[9] An FPTAS has a time complexity that is polynomial in both the problem size and $\frac{1}{\epsilon}$ and produces a solution that is within a factor $(1 + \epsilon)$ of the optimal solution (or $(1 - \epsilon)$ for maximization problems).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2907605, IEEE Internet of Things Journal

Algorithm 1: BDPA $(\mathcal{G}(\mathcal{N},\mathcal{L}),R,\mathcal{F})$

1 $UB \leftarrow +\infty, LB \leftarrow -\infty, \mathcal{C}_p \leftarrow \emptyset, \mathcal{C}_y \leftarrow \emptyset$
2 **while** $UB - LB > \epsilon$ **do**
3 Solve the master problem in Eq. (16) to obtain $Z^{(k)}$, $W^{(k)}$ and $\pi^{(k)}$
4 Let $LB \leftarrow H^{(k)}$
5 Solve the duality of subproblem in Eq. (15) to obtain $U^k(Z^{(k)}, W^{(k)}, \vec{\lambda}^{(k)})$
6 $UB^k \leftarrow \min(UB^{(k-1)}, U^k(Z^{(k)}, W^{(k)}, \vec{\lambda}^{(k)}))$
7 **if** $U^k(Z^{(k)}, W^{(k)}, \vec{\lambda}^{(k)})$ *is bounded* **then**
8 $\mathcal{C}_p$.Add($\vec{\lambda}$) \\ Add extreme point
9 **else**
10 $\mathcal{C}_y$.Add($\vec{\lambda}$) \\ Add extreme ray
11 $k \leftarrow k + 1$
12 Set the cloudlet placement according to $Z^{(k)}$ and $W^{(k)}$.
13 Calculate the task allocation in Eq. (14) to obtain X and Y.

After that, BDPA solves the subproblem based on $Z^{(k)}$ and $W^{(k)}$ returned from solution for solving the master problem in Step 5. The upper bound UB is accordingly updated in Step 6. If $U^k(Z, W, \vec{\lambda})$ is bounded, it indicates that the solution is an extreme point and we will add a proper cut to $\mathcal{C}_p$ in Step 8. Otherwise in Step 10, it implies that the solution is an extreme ray and a respective cut will be added to $\mathcal{C}_y$. The above procedures continue until $UB - LB$ reaches ϵ, which means that a final (optimal) solution is obtained. Consequently, we achieve the cloudlet placement solution according to $Z^{(k)}$ and $W^{(k)}$ in Step 12. In Step 13, we obtain the task allocation solution by solving the LP in Eq. (14) based on the cloudlet placement solution.

Theorem 2: BDPA converges to the optimal solution if we set $\epsilon = 0$.

Proof 2: The proof follows from the correctness and the principle of Bender decomposition algorithm [39]. In general, the optimality gap $UB - LB$ can be calculated at each iteration. Eq. (16) solves the master problem of the original problem and provides the lower bound, while Eq. (15) yields the upper bound of the original problem. In each iteration, Eq. (15) provides a feasible cut constraint for the master problem. Therefore, when $UB - LB = 0$, it indicates that we obtain an optimal solution of the original problem. The proof is therefore complete.

In practice, ϵ is not necessarily set to 0, and during simulations we found that we can still get an optimal or close-to-optimal solution when it is set to a small positive value. This will be reflected in Section VI.

Theorem 3: BDPA terminates in a finite number of iterations for a given ϵ.

Proof 3: Following the proof [41] and according to [42], BDPA will terminate as long as the following conditions are satisfied:

1) The domains of X and Y are non-empty and convex.
2) The objective function of Eq. (8) and constraints in Eqs. (9)-(13) are convex for each fixed Z and W.
3) X and Y are bounded and closed, and the constraint functions are continuous on the domain of X and Y for each fixed Z and W.
4) The original problem with linear relaxation has a finite solution.
5) The original problem with linear relaxation has optimal multipliers λ for the constraints in Eqs. (9)-(13).

It can be seen that Conditions 1, 2 and 3 hold since X and Y are continuous linear variables in a closed range and all the constraints in Eqs. (9)-(13) are linear (and hence convex). Since all the variables are bounded and finite, condition 4 is true. We assume there exists at least one feasible solution to solve the original problem, so the subproblem is feasible. Therefore there exists optimal multipliers λ for the constraints in Eqs. (9)-(13). In all, the proof is complete.

It is worthy to mention that much research have been devoted to exploring ways to shorten the convergence time of the algorithm by reducing the number of iterations and the time required for each iteration. In general, there are four main approaches to deal with this issue [40], namely (1) decomposition strategy, which specifies how the problem is partitioned to obtain the initial master problem and subproblem, (2) The solution procedure, which concerns the algorithms used for the master problem and subproblem, (3) The solution generation, which concerns the method used to set trial values for the complicating variables, and (4) The cut generation, which concerns the strategy used to generate optimality and feasibility cuts. More details can be found in [40] and papers therein.

B. A SDN-based Algorithm Deployment Framework

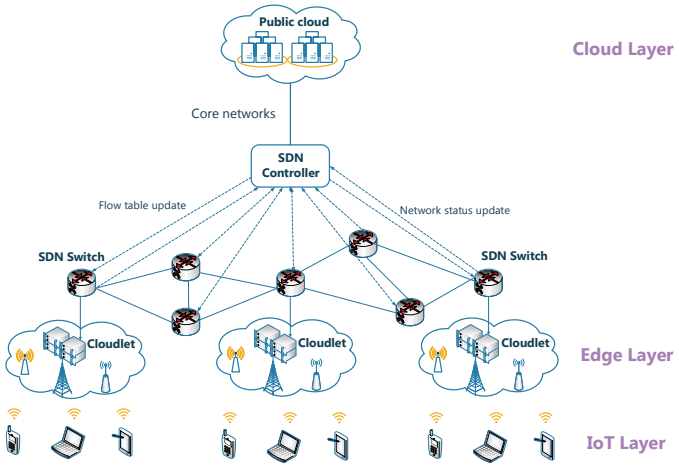


Fig. 4: A SDN-based Algorithm Deployment Framework.

We propose a SDN-based framework to implement and deploy our proposed algorithms. SDN [43] defines a network connection and management methodology that decouples control plane from data plane. In SDN, the network intelligence stays in a logical centralized software-controller (control plane), and network equipments (data plane) can

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2907605, IEEE Internet of Things Journal

be programmed via an open interface (like OpenFlow [44]). According to [45], SDN can benefit MEC and IoT from many areas, e.g., high resolution and effective control, flexibility and low barrier on innovation, service centric implementation, virtual machine mobility, adaptability, low cost solutions, interoperability and Multiplicity of Scope. More specifically, in Fig. 4 each cloudlet is connected with a SDN switch, and all the switches are connected with a central SDN controller. Each SDN switch periodically updates the network status in terms of e.g. node available processing capability to the controller. When a request arrives, the switch first checks its flow table to find any matching rule. If found, it applies the action according to the matched flow rule. If there is an unmatched packet, the switch will forward it to the SDN controller. The SDN controller calculates the solution based on the collected network information for this request according to the proposed solution in Section V. Consequently, the computed solution will be notified and distributed into the SDN switches by updating their flow tables.

VI. Simulation

A. Simulation Setup

We conduct simulations on a network with $V = 30, 40, 50$ nodes, respectively. Each AP node pair is a connected with a link with available bandwidth [5], [17] randomly generated in $[1, 2]$ Gbps[10], and each public cloud and AP node pair is connected with a link with available bandwidth randomly selected in $[0.5, 1]$ Gbps. The cloudlet capacity is set to 300. For each AP node, the number of its direct accessing tasks varies in $[0, 20]$. The aim is to unevenly distribute the tasks to the APs throughout the network. For each request $r(\phi, \Delta, f)$, Δ is chosen in $[20, 40]$ as the delay-sensitive task or in $[70, 120]$ as the delay-tolerant task, f is between $[20, 50]$. The ratio of the number of delay-sensitive tasks and the number of delay-tolerant tasks is $3 : 1$. We vary the total number of task requests from 100 to 300. By doing this, we want to generate heterogeneous task requests. Moreover, for each cloudlet clt, we set $P_o^{clt} = 80$ Watt and $P_t^{clt} = 20$ Watt according to [46]; for each link (i, j), we set $P_o^{ij} = 25$ Watt [47]; for the public cloud, we have $P_t^{cloud} = 30.6$ Watt [34]. For simplicity, we let $\theta_u = \theta_v = 0.1$. We compare our proposed BDPA with two benchmark heuristics as follows:

- Density-Based Clustering (DBC) [14] : DBC first selects one AP n that can issue the biggest number of task requests among all the APs if a cloudlet is placed on it without considering processing capacity. Subsequently, DBC places one cloudlet on n and allocate n to its direct accessing requests one by one without violating the cloudlet capacity. If all its direct accessing tasks are accommodated by the cloudlet on n, and there is still remaining capacity, then n issues the task requests from the other APs one by one ordered by the so-called "relative-distance" without violating its capacity. This procedure continues until all the K cloudlets are placed on the network or all the requests have been issued.
- Randomized Allocation (RA): RA first randomly selects one AP n and places one cloudlet on it. After that, RA allocates the request task directly accessing to n one by one without violating the cloudlet capacity. If all its direct accessing tasks are accommodated by the cloudlet on n and there is still remaining capacity, then n issues the task requests from the other APs without violating its capacity. This procedure continues until all the K cloudlets are placed on the network or all the requests have been issued.

The simulations are run on a high-performance desktop PC with 3.4 GHz and 16 GB memory. We use IBM ILOG CPLEX to implement BDPA and C# to implement the heuristics. We set $\epsilon = 0.05$ in BDPA. Under this setting, we found that the performance of BDPA is the same with the MILP proposed in Section IV-A through simulations in most of the cases, which indicates that BDPA can output the (close-to-)optimal solution. To not clutter the figures, we omit the performances of the exact MILP solution.

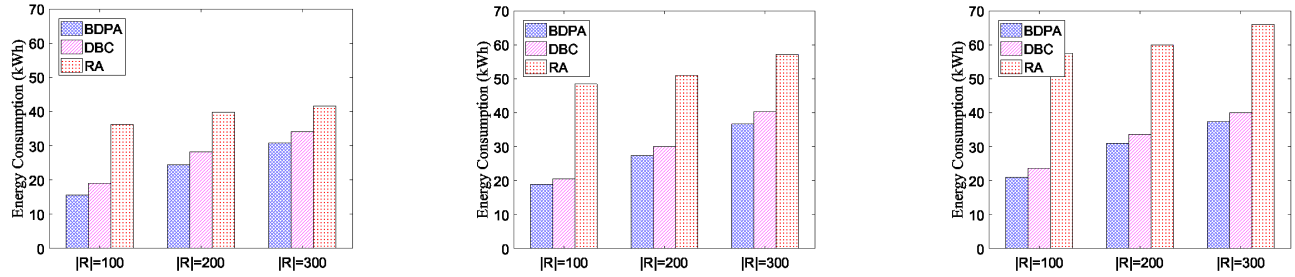
B. Simulation Results

We first compare the energy consumption returned by all these 3 algorithms. Since all the tasks will be processed by the cloudlet and/or the cloud servers and $\theta_u = \theta_v$, the equipment (cloudlet and cloud server) processing energy consumption for all the tasks returned by all these 3 algorithms are the same. We therefore omit the processing energy consumption value and only compare the idle energy consumption of cloudlets and links. We set $K = V - 1$ for DBC and RA. As such, Fig. (5) plots the energy consumption returned by all these algorithms for different number of network nodes. We see that for all the algorithms, their returned energy consumption value increases when the number of task requests raises. This is because more cloudlets and links need to be placed and switched on in order to process the required tasks, which in turns consume more energy. Nevertheless, the proposed BDPA can always return the lowest energy consumption value, followed by the DBC and RA performs the worst. We also notice that for the same number of task requests, the energy consumption value returned by RA increases with N increasing, but the energy consumption values returned by BDPA and DBC keep similar. This is due to the fact that RA always randomly selects one node to place the cloudlet, but the AP node may be in charge of small number of or no task requests, which means that the cloudlet cannot be always well utilized if the other requests cannot be processed on it because of the delay violation. Moreover, since we set $K = V - 1$, as long as there are still remaining task requests to accommodate, RA will place a new cloudlet on the node (and switch on new links if necessary) to accommodate these task requests, which consumes more energy consumption as N increases. Tables II-IV give the number of placed cloudlets and switched-on links for all the algorithms.

After that, we compare the algorithms in terms of Acceptance Ratio (AR), which is defined as the number of accepted

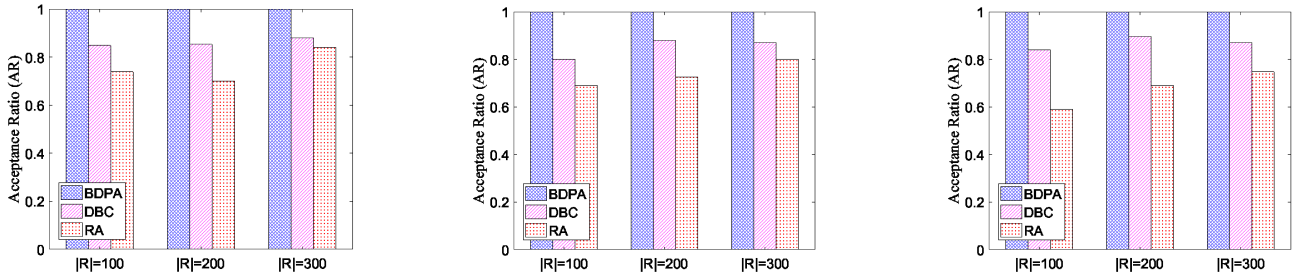
[10]The APs can be connected with either wired or wireless links, which depends on the specific scenario. Following [5], [17], we assume that all APs are connected by wired links in the context of SDN-based cellular core/IoT networks.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2907605, IEEE Internet of Things Journal



(a) $V = 30$ (b) $V = 40$ (c) $V = 50$

Fig. 5: Energy Consumption for different number of total nodes (V): (a) $V = 30$ (b) $V = 40$ (c) $V = 50$.



(a) $V = 30$ (b) $V = 40$ (c) $V = 50$

Fig. 6: Acceptance Ratio for different number of total nodes (V): (a) $V = 30$ (b) $V = 40$ (c) $V = 50$.

task requests over the total number of task requests. In order to have a fair comparison, we set $K = 6, 12, 18$ for DBC and RA when $V = 30, 40, 50$, respectively, which are the number of needed cloudlets in the optimal solutions. By doing this, we constrain the maximum allowable number of placed cloudlets for DBC and RA as the same of the optimal solution. Under this circumstance, Fig. (6) illustrates the AR value of all the algorithms. We see that BDPA can always accept all the task requests, which also verifies its correctness. DBC obtains the second highest AR value and RA achieves the lowest AR value. It implies that DBC and RA as heuristics still fail to accommodate a certain amount of task requests and hence cannot be efficiently used for multiple task requests in a resource-constrained scenario.

Finally, Fig. 7 provides the iterations needed for BDPA to converge. We see that with V and $|R|$ increasing, the number of iterations also increases. Nevertheless, BDPA can always converge in a finite number of iterations, which proves the correctness of Theorem 3.

TABLE II: Number of cloudlets and links used by different algorithms for $V = 30$.

	BDPA		DBA		RA	
Task	Cloudlets	Links	Cloudlets	Links	Cloudlets	Links
100	6	33	8	38	28	31
200	12	43	15	46	29	40
300	18	45	23	40	29	46

TABLE III: Number of cloudlets and links used by different algorithms for $V = 40$.

	BDPA		DBA		RA	
Task	Cloudlets	Links	Cloudlets	Links	Cloudlets	Links
100	6	44	8	43	39	37
200	12	53	14	55	39	45
300	18	65	21	67	39	66

TABLE IV: Number of cloudlets and links used by different algorithms for $V = 50$.

	BDPA		DBA		RA	
Task	Cloudlets	Links	Cloudlets	Links	Cloudlets	Links
100	6	51	8	53	45	38
200	12	65	14	67	48	56
300	18	68	22	63	49	67

VII. Conclusion

In this paper, we have first investigated how to calculate the task completion delay and presented the energy consumption models of different equipments in MEC. After that, we have studied the Cloudlet Placement and Task Allocation (CPTA) problem, which is to place cloudlets on AP nodes and allocate each user's tasks to corresponding cloudlets and public cloud, such that the total energy consumption is minimized and each task's delay requirement is satisfied. We have shown this problem is NP-hard and proposed a Benders Decomposition-based algorithm to solve it. We have also presented a SDN-based

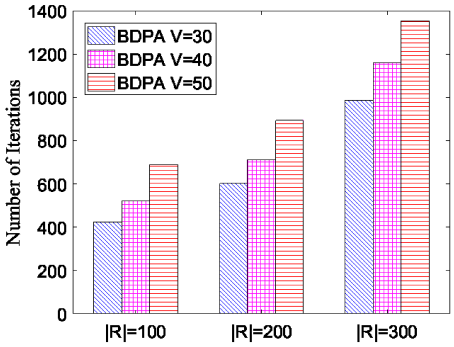


Fig. 7: Number of Iterations for BDPA.

framework to deploy our proposed algorithm. The simulations reveal that the proposed algorithm can achieve an optimal or close-to-optimal performance in terms of energy consumption and acceptance ratio compared with two benchmark heuristics.

In this paper, we do not consider queueing tasks in the cloudlet, i.e., we assume no buffer is available for queueing the tasks, similar to [25]. The reason is that queueing tasks is related to the buffer size [48], [49] of cloudlet. If the buffer size is large enough, all the tasks can be queued but this case is not very practical since it is too costly. If the buffer size is less than the size of to be queued tasks, this deals with which tasks need to be queued and which tasks need to be relayed. In this sense, taking the buffer size for queueing tasks into account will make the problem more difficult to solve. We will further explore the scenario of queueing tasks in the cloudlet in the CPTA problem in our future work.

ACKNOWLEDGMENT

The work of Song Yang is partially supported by the National Natural Science Foundation of China (NSFC, No. 61802018) and Beijing Institute of Technology Research Fund Program for Young Scholars. The work of Fan Li is partially supported by the NSFC (No. 61772077, 61370192), and the Beijing Natural Science Foundation (No. 4192051). The work of Meng Shen is partially supported by the National Key Research and Development Program of China (No. 2018YFB0803405), the NSFC (No. 61602039), the Beijing Natural Science Foundation (No. 4192050), and CCF-Tencent Open Fund WeBank Special Funding. The work of Xu Chen is partially supported by the NSFC (No. U1711265) and the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (No. 2017ZT07X355). The work of Xiaoming Fu is partially supported by EU FP7 CleanSky ITN (No. 607584) and H2020 RISE COSAFE (No. 824019) projects. The work of Yu Wang is partially supported by the NSFC (No. 61572347) and the US Department of Transportation Center for Advanced Multimodal Mobility Solutions and Education (No. 69A3351747133). Song Yang is the corresponding author.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *Communications of the ACM*, vol. 53, no. 6, p. 50, 2010.

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

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," *ETSI White paper*, 2015.

[4] D. Greenfield, "Fog computing vs. edge computing: What is the difference?" 2016. [Online]. Available: https://www.automationworld.com/fog-computing-vs-edge-computing-whats-difference

[5] X. Sun and N. Ansari, "Green cloudlet network: A sustainable platform for mobile cloud computing," *IEEE Transactions on Cloud Computing*, 2018.

[6] X. Li, D. Li, J. Wan, C. Liu, and M. Imran, "Adaptive transmission optimization in sdn-based industrial internet of things with edge computing," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1351–1360, 2018.

[7] M. Shen, B. Ma, L. Zhu, X. Du, and K. Xu, "Secure phrase search for intelligent processing of encrypted data in cloud-based iot," *IEEE Internet of Things Journal*, 2019.

[8] L. Rao, X. Liu, M. D. Ilic, and J. Liu, "Distributed coordination of internet data centers under multiregional electricity markets," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 269–282, 2012.

[9] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[11] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[12] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 1826–1857, 2018.

[13] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 416–464, 2018.

[14] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, Oct 2017.

[15] M. Jia, W. Liang, Z. Xu, M. Huang, and Y. Ma, "Qos-aware cloudlet load balancing in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, 2018.

[16] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, 2017, pp. 1–10.

[17] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal placement of cloudlets for access delay minimization in SDN-based internet of things networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1334–1344, 2018.

[18] L. Ma, J. Wu, and L. Chen, "DOTA: Delay bounded optimal cloudlet deployment and user association in wmans," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2017, pp. 196–203.

[19] J. Meng, W. Shi, H. Tan, and X. Li, "Cloudlet placement and minimum-delay routing in cloudlet computing," in *IEEE International Conference on Big Data Computing and Communications (BIGCOM)*, 2017, pp. 297–304.

[20] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, 2018.

[21] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2866–2880, Oct 2016.

[22] Q. Fan and N. Ansari, "Cost aware cloudlet placement for big data processing at the edge," in *IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[23] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *IEEE INFOCOM*, 2017, pp. 1–9.

[24] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[25] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[26] L. Chen, S. Liu, B. Li, and B. Li, "Scheduling jobs across geo-distributed datacenters with max-min fairness," in *IEEE INFOCOM*, 2017, pp. 1–9.

[27] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, 2015, pp. 421–434.

[28] E. Meskar and B. Liang, "Fair multi-resource allocation with external resource for mobile edge computing," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 184–189.

[29] M. Salmani and T. N. Davidson, "Uplink resource allocation for multiple access computational offloading," *arXiv preprint arXiv:1809.07453*, 2018.

[30] ——, "Multiple access computational offloading: Communication resource allocation in the two-user case," *arXiv preprint arXiv:1805.04981*, 2018.

[31] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[32] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, 2018.

[33] M. Xia, M. Tornatore, Y. Zhang, P. Chowdhury, C. U. Martel, and B. Mukherjee, "Green provisioning for optical wdm networks," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 17, no. 2, pp. 437–445, 2011.

[34] S. Yang, P. Wieder, R. Yahyapour, and X. Fu, "Energy-aware provisioning in optical cloud networks," *Computer Networks*, vol. 118, pp. 78 – 95, 2017.

[35] J. Buysse, K. Georgakilas, A. Tzanakaki, M. De Leenheer, B. Dhoedt, and C. Develder, "Energy-efficient resource-provisioning algorithms for optical clouds," *Journal of Optical Communications and Networking*, vol. 5, no. 3, pp. 226–239, 2013.

[36] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the internet with nano data centers," in *Proceedings of the 5th ACM international conference on Emerging networking experiments and technologies*, 2009, pp. 37–48.

[37] F. Giroire, J. Moulierac, T. K. Phan, and F. Roudaut, "Minimization of network power consumption with redundancy elimination," *Computer communications*, vol. 59, pp. 98–105, 2015.

[38] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman & Co., 1979.

[39] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[40] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei, "The benders decomposition algorithm: A literature review," *European Journal of Operational Research*, vol. 259, no. 3, pp. 801–817, 2017.

[41] P. Zhao and G. Dn, "A benders decomposition approach for resilient placement of virtual process control functions in mobile edge clouds," *IEEE Transactions on Network and Service Management*, 2018.

[42] C. Floudas, *Nonlinear and mixed-integer optimization: funmamentals and applications*. Oxford University Press, 1995.

[43] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[44] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

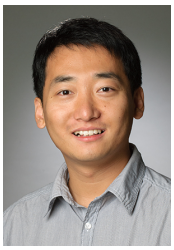
[45] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[46] Q. Fan, N. Ansari, and X. Sun, "Energy driven avatar migration in green cloudlet networks," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1601–1604, 2017.

[47] W. V. Heddeghem, M. D. Groote, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester, "Energy-efficiency in telecommunications networks: Link-by-link versus end-to-end grooming," in *IEEE 14th Conference on Optical Network Design and Modeling (ONDM)*, 2010, pp. 1–6.

[48] B. Steyaert, H. Bruneel, and Y. Xiong, "A general relationship between buffer occupancy and delay in discrete-time multiserver queueing models, applicable in atm networks," in *IEEE INFOCOM*, 1993, pp. 1250–1258.

[49] B. N. Vellambi, N. Torabkhani, and F. Fekri, "Throughput and latency in finite-buffer line networks," *IEEE Transactions on Information Theory*, vol. 57, no. 6, pp. 3622–3643, 2011.



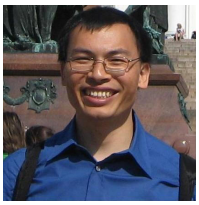
Song Yang received the Ph.D. degree from Delft University of Technology, The Netherlands, in 2015. From August 2015 to July 2017, he worked as postdoc researcher for the EU FP7 Marie Curie Actions CleanSky Project in Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Göttingen, Germany. He is currently an associate professor at School of Computer Science and Technology in Beijing Institute of Technology, China. His research interests focus on data communication networks, cloud/edge computing, and network function virtualization.

Fan Li is a Professor at the School of Computer Science and Technology in Beijing Institute of Technology, China. She received her PhD degree in Computer Science from the University of North Carolina at Charlotte, MEng degree in Electrical Engineering from the University of Delaware, MEng and BEng degrees in communications and information system from Huazhong University of Science and Technology, China, respectively. Her current research focuses on wireless networks, smart sensing, crowd sensing and mobile computing. Her papers won Best Paper Awards from IEEE MASS (2013), IEEE IPCCC (2013), ACM MobiHoc (2014), and Tsinghua Science and Technology (2015). She is a member of ACM and IEEE.

Meng Shen received the B.Eng degree from Shandong University, Jinan, China in 2009, and the Ph.D degree from Tsinghua University, Beijing, China in 2014, both in computer science. Currently he serves in Beijing Institute of Technology, Beijing, China, as an associate professor. His research interests include privacy protection for cloud and IoT, blockchain applications, and encrypted traffic classification. He received the Best Paper Runner-Up Award at IEEE IPCCC 2014. He is a member of IEEE.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2907605, IEEE Internet of Things Journal

Xu Chen is a Full Professor with Sun Yat-sen University, Guangzhou, China, and the vice director of National and Local Joint Engineering Laboratory of Digital Home Interactive Applications. He received the Ph.D. degree in information engineering from the Chinese University of Hong Kong in 2012, and worked as a Postdoctoral Research Associate at Arizona State University, Tempe, USA from 2012 to 2014, and a Humboldt Scholar Fellow at Institute of Computer Science of University of Goettingen, Germany from 2014 to 2016. He received the prestigious Humboldt research fellowship awarded by Alexander von Humboldt Foundation of Germany, 2014 Hong Kong Young Scientist Runner-up Award, 2016 Thousand Talents Plan Award for Young Professionals of China, 2017 IEEE Communication Society Asia-Pacific Outstanding Young Researcher Award, 2017 IEEE ComSoc Young Professional Best Paper Award, Honorable Mention Award of 2010 IEEE international conference on Intelligence and Security Informatics (ISI), Best Paper Runner-up Award of 2014 IEEE International Conference on Computer Communications (INFOCOM), and Best Paper Award of 2017 IEEE Intranational Conference on Communications (ICC). He is currently an Associate Editor of IEEE Internet of Things Journal and IEEE Journal on Selected Areas in Communications (JSAC) Series on Network Softwarization and Enablers.



Xiaoming Fu received his Ph.D. in computer science from Tsinghua University, Beijing, China in 2000. He was then a research staff at the Technical University Berlin until joining the University of Göttingen, Germany in 2002, where he has been a professor in computer science and heading the Computer Networks Group since 2007. He has spent research visits at universities of Cambridge, Uppsala, UPMC, Columbia, UCLA, Tsinghua, Nanjing, Fudan, and PolyU of Hong Kong. Prof. Fu's research interests include network architectures, protocols, and applications. He is currently an editorial board member of IEEE Communications Magazine, IEEE Transactions on Network and Service Management, and Elsevier Computer Communications, and has served on the organization or program committees of leading conferences such as INFOCOM, ICNP, ICDCS, MOBICOM, MOBIHOC, CoNEXT, ICN and COSN. He is an IEEE Senior Member and an IEEE Communications Society Distinguished Lecturer.

Yu Wang is a Professor of Computer Science at the University of North Carolina at Charlotte. He received his Ph.D. degree in Computer Science from Illinois Institute of Technology, his B.Eng. degree and M.Eng. degree in Computer Science from Tsinghua University, China. His research interest includes wireless networks, smart sensing, mobile computing, and algorithm design. His research has been continuously supported by federal agencies including US National Science Foundation, US Department of Transportation, and National Natural Science Foundation of China (NSFC). He has published over 150 papers in peer reviewed journals and conferences, with four best paper awards. He is a recipient of Ralph E. Powe Junior Faculty Enhancement Awards from Oak Ridge Associated Universities (2006), Outstanding Faculty Research Award from College of Computing and Informatics at UNC Charlotte (2008), and Overseas Young Scholars Cooperation Research Fund from NSFC (2014). He is a senior member of the ACM and a fellow of the IEEE.