Towards Revenue-Driven Multi-User Online Task Offloading in Edge Computing

Zhi Ma, Sheng Zhang, *Member, IEEE*, Zhiqi Chen, Tao Han, *Member, IEEE*, Zhuzhong Qian, *Member, IEEE*, Mingjun Xiao, *Member, IEEE*, Ning Chen, Jie Wu, *Fellow, IEEE*, and Sanglu Lu, *Member, IEEE*

Abstract—Mobile Edge Computing (MEC) has become an attractive solution to enhance the computing and storage capacity of mobile devices by leveraging available resources on edge nodes. In MEC, the arrivals of tasks are highly dynamic and are hard to predict precisely. It is of great importance yet very challenging to assign the tasks to edge nodes with guaranteed system performance. In this article, we aim to optimize the revenue earned by each edge node by optimally offloading tasks to the edge nodes. We formulate the revenue-driven online task offloading (ROTO) problem, which is proved to be NP-hard. We first relax ROTO to a linear fractional programming problem, for which we propose the Level Balanced Allocation (LBA) algorithm. We then show the performance guarantee of LBA through rigorous theoretical analysis, and present the LB-Rounding algorithm for ROTO using the primal-dual technique. The algorithm achieves an approximation ratio of $2(1 + \xi) \ln(d + 1)$ with a considerable probability, where d is the maximum number of process slots of an edge node and ξ is a small constant. The performance of the proposed algorithm is validated through both trace-driven simulations and testbed experiments. Results show that our proposed scheme is more efficient compared to baseline algorithms.

---

1 INTRODUCTION

Nowadays, pervasive mobile computing and the Internet of Things are driving the development of many new compute-intensive and latency-sensitive applications, such as mobile gaming and virtual/augmented reality (VR/AR), and massive data will be generated at the edge of networks. However, many devices, such as smartphones and wearable devices, have a limited processing capacity and may not be able to process their data. Due to network bandwidth, storage and data privacy concerns, it is also impractical, and often unnecessary, to send all of the data to a remote cloud. Fortunately, Mobile-Edge Computing (MEC) has been gaining strong momentum as an emerging paradigm that provides cloud computing-like capabilities including computing and storage resources, at the edge of wireless access networks. Although MEC is less powerful than a remote cloud [1], [2], the transmission latency between a user and a mobile edge cloud is much lower than that of the remote cloud as it is located at the network edge.

In a real-world task-offloading scenario, the arrivals of users are dynamic and the tasks must be processed quickly. This motivates us to consider the online scenario where users arrive dynamically, and resources are allocated based on only the past offloading decisions and current states of the edge nodes. Therefore, making correct decisions when the task arrivals are uncertain is challenging. Meanwhile, with the increasing complexity of applications and wireless networks, the scale of the dynamic offloading problem is an enormous obstacle [3].

To tackle this issue, Li *et al.* [4] proposed an online computation rate maximization algorithm using the Lyapunov method for a multi-user MEC system by jointly managing the radio and computational resources and allocating time for energy transfer and data transmission; Chen *et al.* [5] formulated a multi-user multi-task computation offloading problem for green MEC and used the Lyapunov optimization approach to determine the energy harvesting policy. Although the Lyapunov optimization is often used to deal with online problems, it takes time to converge, and cannot fully adapt to bursts of task requests in a short time. Some other studies chose to use deep learning or deep Q-network (DQN) methods [6], [7], [8], [9], [10], [11], [12], which require training in advance and have no explicit theoretical performance guarantees.

Meanwhile, these studies mainly focused on the energy-efficient and resource-efficient computational service offloading scheme in MEC. Only a few works have considered

- *Zhi Ma, Sheng Zhang, Zhiqi Chen, Zhuzhong Qian, Ning Chen, and Sanglu Lu are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210023, China. E-mail: marszer@foxmail.com, {sheng, qzz, sanglu}@nju.edu.cn, cszczq123456@gmail.com, ningc@smail.nju.edu.cn.*
- *Tao Han is with the Department of the Electrical and Computer Engineering, University of New Jersey Institute of Technology, Newark, NJ 07102 USA. E-mail: Tao.Han@uncc.edu.*
- *Mingjun Xiao is with the School of Computer Science and Technology / Suzhou Institute for Advanced Study, University of Science and Technology of China, Hefei, Anhui 230052, China. E-mail: xiaomj@ustc.edu.cn.*
- *Jie Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA. E-mail: jiewu@temple.edu.*
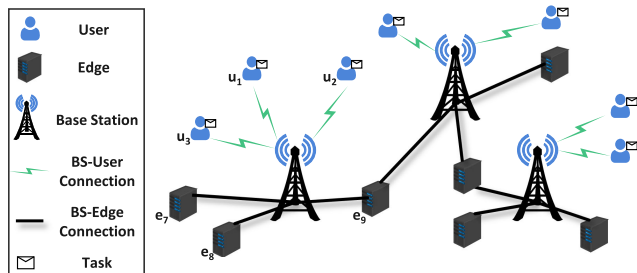
Fig. 1. System illustration. Multiple edge nodes constitute a mobile edge cloud. We aim to optimize the revenue earned by the edge nodes.

the revenue maximization problem in task offloading [13], [14], [15], [16], and these works have considered either the edge offline scenario or cloud-edge competition scenario.

Service providers such as Amazon and Alibaba have deployed many edge nodes and built their own edge node server platforms, which can provide the computing power of content delivery network (CDN) edge nodes (ENS [17] and Lambda@edge[18]). Enterprises or individuals can rent their edge nodes by paying on demand or on time. In this work, we study the revenue-driven online task offloading (ROTO) problem in MEC. Specifically, we focus on offloading multiple computation intensive tasks to a set of edge nodes, and the objective is to maximize the revenue of edge nodes from the perspective of service providers. Fig. 1 illustrates a typical offloading scenario in which a set of edge nodes constitute a mobile edge cloud. Generally, edge node services are constructed based on the operator's network and the task offloading is done by the Base Stations (BS) of the operators. We assume that the service provider can obtain the information of the BS and control some functions of the BS through software define network (SDN). Users can offload their tasks to the edge nodes to extend their own computing ability via paying for task execution. Moreover, a user can offload its task only to edge nodes if the user and edge nodes are within the communication range of the same BS. A user first sends its task to the nearby BS, then the BS decides how to offload the task to the edge nodes within its communication range according to the revenue, computing demand of the task, and states of the edge nodes.

We first formulate the ROTO problem into an integer linear programming problem, then we prove that ROTO is NP-complete by reducing the multi-dimensional knapsack problem to it. To solve it, we first relax it to a linear fractional programming problem, i.e., ROTO-LP. We then present two important notions, *level* and *move-up energy*, that enable us to design an efficient algorithm, named level balanced allocation (LBA). By intelligently constructing a potential function and using the primal-dual schema, we prove that LBA can achieve an approximation ratio of $2(1 + \xi) \ln(d + 1)$, in which d is the maximum number of the process slots of an edge node. Based on the intuitions obtained from designing LBA for ROTO-LP, we finally propose the level balanced rounding (LBR) algorithm for ROTO by combining LBA and the rounding technique. Using the Chernoff bound and probability analysis, we prove LBR can achieve the same approximation ratio as LBA with a high probability, i.e., at least $1 - e^{-\sigma n}$, where n is the number of users. We implemented LBR on our testbed consisting of 8 Raspberry

Pis and 4 mobile phones. Trace-driven simulations and testbed experiments reveal the effectiveness of LBR.

Our main contributions are summarized as follows:

- We develop a multi-user computation offloading framework for a mobile edge computing system to maximize the total revenue. We provide a formal formulation of the revenue-driven online task offloading (ROTO) problem, which proved to be NP-complete.
- We design the Level Balanced Allocation (LBA) algorithm to solve ROTO-LP, which achieves an approximation ratio of $2(1 + \xi) \ln(d + 1)$.
- Based on LBA, we propose the Level Balanced Rounding (LBR) algorithm and obtain the solution of ROTO. We prove that LBR achieves an approximation ratio of $2(1 + \xi) \ln(d + 1)$ with a probability of at least $1 - e^{-\sigma n}$.
- We conduct trace-driven simulations and testbed experiments to evaluate the performance of the proposed algorithm. The results are shown from different perspectives to provide conclusions.

The rest of the paper is organized as follows. Prior works are reviewed in Section 2. The system model is discussed and the offloading problem is formulated in Section 3. The proof of the ROTO NP-hardness is also explained in Section 3. The online algorithm to provide maximum revenue is proposed and analyzed in Section 4. Simulation results and testbed experiments are investigated in Sections 5 and 6, respectively. In Section 7, we conclude the paper.

2 RELATED WORK

In this section, we give a brief overview about some related works in regards to task offloading in mobile edge computing.

First of all, task offloading is divided into two categories according to whether there are dependencies between tasks. The problem of offloading dependent tasks in MEC is complicated thus most of works will make many assumptions. Kao *et al.* [19] concerned a dependent task assignment problem over multiple devices. However, they did not impose restrictions on the capacity of the devices, which makes their algorithm more inclined to offload the tasks on a few devices with more capable devices. GenDoc [20] jointly considered the problem of dependent task offloading and service caching placement with the objective of application completion time minimization. However, GenDoc does not consider the computing capacities when offloading tasks to edge nodes. In fact, mobile edge nodes are resource-sensitive and GenDoc may cause irrational use of limited computing resources. In the field of offloading independent tasks, Jošilon *et al.* [21] used game theory to coordinate offloading various task requests from multi-user to the mobile edge cloud. Zhu *et al.* [22] investigated the task offloading problem in wireless powered mobile edge computing. Zhao *et al.* and Ma *et al.* [23], [24] considered the factor of service caching when offloading tasks. Chen *et al.* [25] leveraged the idea of software defined network, and investigated the task offloading problem in ultra-dense networks. Tao *et al.* and Jošilo *et al.* [26], [27] focused on task offloading of

autonomous devices. In this paper, we focus on offloading the independent tasks in MEC.

The above works considered offline scenarios only, that is, the arrival of all tasks is known in advance. However, to further enhance the agility of the mobile edge cloud, an online algorithm is more preferred. When considering online offloading scenarios, most studies chose to use deep learning [6], [7] or reinforcement learning methods [9], [10], [11] for different objectives such as maximizing the weighted sum computation rate [9], minimizing the energy consumption [6] and minimizing the running cost [7], [10], [11]. However, these works require training in advance and have no explicit theoretical performance guarantee. In addition to machine learning-related methods, Lyapunov optimization approach is often applied to analyze the online offloading. Chen *et al.* [5] discussed multi-user multi-task offloading scheduling schemes in a renewable mobile edge cloud system, and used Lyaponov optimization approach to determine the energy harvesting policy. Ning *et al.* [28] comprehensively consider both of MEC and could computing and design an iterative heuristic algorithm to minimize the offloading delay. Guo *et al.* [29] investigated the problem of collaborative mobile-edge computation offloading in 5G HetNets and proposed a game-theoretical computation offloading scheme. However, these studies mainly focused on the energy-efficient and resource-efficient computation offloading scheme in MEC. Different from theirs, we consider how to maximize the benefits of edge nodes from the perspective of service providers. In additional, Lyapunov method takes time to converge, and cannot fully adapt to bursts of task requests in a short time. We use the primal-dual theory instead of deep learning or Lyapunov's theorem to solve the problem. What's more, in solving this problem, we propose our algorithm with theoretical guarantees.

3 MODEL AND PROBLEM FORMULATION

In this section, we first elaborate the computation task model and edge node execution model. Then we formulate task computation offloading as an optimization problem to maximize offloading revenue. The goal of the optimization is to determine the optimal edge nodes to offload tasks for users based on the arrival of users and their resource requirements. For ease of presentation, we only consider the CPU resource. Other type of resources such as memory usage or disk I/O cycles can be similarly addressed [19], [30]. After that, we provide the proof of the ROTO NP-hardness.

3.1 Computation Task Model

Consider a time horizon $\mathbb{T}$, which for simplicity is taken to be discrete but can be extended to be continuous. We assume that there are n independent users that need to offload their tasks to edge nodes, where the set of users is denoted as $\mathbb{U} = \{u_1, u_2, \ldots, u_n\}$. The task of user u_i is characterized by a set of parameters, $\{S_i, \beta_i, T_i^{arr}, T_i^{ddl}, \alpha_{i1}, \ldots, \alpha_{ij}, \ldots, \alpha_{im}\}$, in which S_i denotes the input data size (in byte); β_i denotes the number of CPU cycles required to process one byte of data [31]; T_i^{arr} and T_i^{ddl} represent the arrival time and deadline of the task, respectively; α_{ij} represents the ratio between the revenue that edge node e_j can get and

the computing demand of u_i [5], [32]. Note that, if e_j and u_i are not within the communication range of the same BS, we set α_{ij} as zero.

Considering the rapid development of 5G networks and short distance between users and edge nodes, we assume that the time of data transmission between a user and an edge node is small enough to be ignored [33]. Thus, the required computing demand (CPU cycles per second) of u_i is equal to $(\beta_i \cdot S_i)/(T_i^{ddl} - T_i^{arr})$, which is denoted as b_i. Based on this, we define the revenue that edge node e_j can get if the task of u_i is offloaded to e_j as $\alpha_{ij}b_i$.

According to our survey, the price paid by the user is based on the computation resource used, service type and service time slot [34], [35], [36]. We focus on compute intensive tasks, which means the service types of all tasks are the same. And the price for running the same task on different edge nodes will not vary greatly. Thus we can assume that $\max_{e_j \in C(i)} \alpha_{ij} \leq (1 + \xi)\min_{e_j \in C(i)} \alpha_{ij}$, where ξ is a small constant and $C(i)$ is the set of edge nodes that can provide servers to u_i.

In this paper, we focus on the scenarios in which the mobile device has very limited computational resources, and hence all the tasks should be offloaded to edge nodes for execution. The task of each user is atomic and cannot be further divided, and one task is assumed to be offloaded to only one edge node.

3.2 Edge Node Execution Model

We assume that there are m independent edge nodes that can provide computing service for users, where the set of edge nodes is denoted as $\mathbb{E} = \{e_1, e_2, \ldots, e_m\}$. Edge node e_j has V_j process slots, which means e_j can process at most V_j tasks at the same time, and these process slots share the entire computing capacity. Denote y_{ij} as a binary indictor: $y_{ij} = 1$ if the task of u_i is offloaded to edge node e_j and $y_{ij} = 0$ otherwise. Hence

$$\forall \, e_j \in \mathbb{E} : \sum_{i=1}^{n} y_{ij} \leq V_j, \tag{1}$$

$$\forall \, u_i \in \mathbb{U}, e_j \in \mathbb{E} : y_{ij} \in \{0, 1\}. \tag{2}$$

A user can connect to an edge node if the user and edge node are within the communication range of the same BS. Denote $C(i)$ as the set of edge nodes that are within the communication range of the same BS as u_i. To meet the task offloading feasibility constraints, it requires

$$\forall \, u_i \in \mathbb{U} : \sum_{e_j \in C(i)} y_{ij} \leq 1, \tag{3}$$

$$\forall \, u_i \in \mathbb{U}, e_j \notin C(i) : y_{ij} = 0. \tag{4}$$

Denoting the computing capacity of edge node e_j as B_j (CPU cycles per second). An edge node cannot provide computing resources that exceed its capacity. Hence

$$\forall \, e_j \in \mathbb{E}: \qquad \sum_{i=1}^{n} b_i y_{ij} \leq B_j. \tag{5}$$

In this paper we assume that the computing demand b_i is not less than $\min_{e_j \in C(i)} B_j/V_j$. This assumption is reasonable,

because there is no need for users to offload their task to edge nodes for processing if the computing demand is small.

3.3 Problem Formulation

In this work, we intend to maximize the overall revenue of edge nodes. The main problem studied is defined as follows:

Problem 1. *Given a time horizon* $\mathbb{T}$*, a set of users* $\mathbb{U}$ *with parameters* $\{S_i, \beta_i, T_i^{arr}, T_i^{ddl}, \alpha_{i1}, \ldots, \alpha_{ij}, \ldots, \alpha_{im}\}$ *corresponding to the task of* u_i*, a set of edge nodes* $\mathbb{E}$ *with parameters* $\{B_j, V_j\}$ *corresponding to* e_j*, and* n *sets* $C(1), \ldots, C(n)$ *denoting the edge nodes that are connected to* u_i*, the revenue-driven online task offloading problem is to find an allocation of tasks to edge nodes that maximizes the overall revenue.*

$$\max \sum_{i=1}^{n} \sum_{e_j \in C(i)} \alpha_{ij} b_i y_{ij}, \tag{6}$$

$$s.t. (1) - (5), \tag{7}$$

where $b_i = (\beta_i S_i)/(T_i^{ddl} - T_i^{arr})$*.*

Theorem 1. *The ROTO is NP-complete.*

Proof. We prove this by reducing the multi-dimensional knapsack problem to ROTO, which is NP-complete. The multi-dimensional knapsack problem is defined as follows: the weight of knapsack item i is given by a D-dimensional vector $\overline{w_i} = \{w_{1i}, \ldots, w_{iD}\}$ and the knapsack has a D-dimensional capacity vector $\{W_1, \ldots, W_D\}$. Knapsack item i has its value v_i. The target is to maximize the sum of values of items in the knapsack so that the sum of the weight in each dimension j does not exceed W_j.

We construct the tasks and edge nodes as follows:

- We set the number of edge node to be 1, the process slot to be V and the process capacity to be B.
- For each item i in knapsack, we construct a task u_i in ROTO,
- For the weight of item i, we construct a vector $\{1, b_i\}$ in ROTO,
- For the value of item i, we construct the reward $\alpha_i b_i$ of u_i in ROTO.
- For the capacity of knapsack, we construct a vector $\{V, B\}$ in ROTO.

The construction can be finished in polynomial time; thus, we reduce solving the NP-complete multi-dimensional knapsack problem to solving a special case of ROTO, implying that ROTO is NP-complete.

With this transformation, we can prove the theorem easily: assume, without loss of generality, there is a solution to the multi-dimensional knapsack problem, then this solution is also the answer to the special case of ROTO. □

4 ALGORITHMIC DESIGN

In this section, we first relax ROTO to a linear fractional programming problem (Section 4.1), for which we propose the LBA algorithm based on the level and move-up energy notions (Section 4.2). We then show the performance

guarantee of LBA through rigorous theoretical analysis (Section 4.3), after which we present the LBR algorithm for ROTO and prove it has the same approximation ratio as LBA with a high probability (Section 4.4).

4.1 Relaxing ROTO to ROTO-LP

We relax constraint (2) and get the following linear formulation of ROTO, which we call ROTO-LP:

$$\max \sum_{i=1}^{n} \sum_{e_j \in C(i)} \alpha_{ij} b_i y_{ij} \tag{8}$$

$$s.t. \quad \forall\, u_i \in \mathbb{U}: \qquad \sum_{e_j \in C(i)} y_{ij} \leq 1, \tag{9}$$

$$\forall\, u_i \in \mathbb{U}, e_j \notin C(i): \qquad y_{ij} = 0, \tag{10}$$

$$\forall\, e_j \in \mathbb{E}: \qquad \sum_{i=1}^{n} b_i y_{ij} \leq B_j, \tag{11}$$

$$\forall\, e_j \in \mathbb{E}: \qquad \sum_{i=1}^{n} \lceil y_{ij} \rceil \leq V_j, \tag{12}$$

$$\forall\, u_i \in \mathbb{U}, e_j \in \mathbb{E}: \qquad y_{ij} \in [0, 1]. \tag{13}$$

Note that it is necessary to round y_{ij} in constraint (12), which makes the result meet the constraint that edge node e_j can process at most V_j tasks at the same time.

For an online setting, at each time $t \in \mathbb{T}$, only a task of user $u_i \in \mathbb{U}$ that arrives before t is known. This implies both the objective function (8) and the left hand side sum in Eqs. (9), (11) and (12) are unknown ahead, and they are gradually revealed to the algorithm over the operation period. The online algorithm does not know the length of $\mathbb{T}$ and has to take into account possible future arrivals and reserve the resources properly.

4.2 The Online Algorithm for ROTO-LP

In this subsection, we propose LBA to solve ROTO-LP, in which each task can be divided into arbitrary size.

4.2.1 Preliminaries

Let d be the maximum number of process slots of all edges, i.e., $d = \max_j V_j$. Denote the amount of computing resources e_j has allocated to users by Ω_j. The notion of the level is defined as follows.

Definition 1. *(Level) Each edge node* e_j *is associated with a level* L_j*, which is an integer between* $(d - V_j)$ *and* d*. The level of an edge node changes only when the amount of computing resources it has allocated to users changes. Formally, we have*

$$L_j \triangleq d - V_j + \left\lfloor \frac{V_j \Omega_j}{B_j} \right\rfloor. \tag{14}$$

From the above definition, we know if $\Omega_j = 0$, then $L_j = d - V_j$; if $\Omega_j = B_j$, then $L_j = d$. We design this notion, level, for capturing how 'full' an edge node is. The proposed LBA
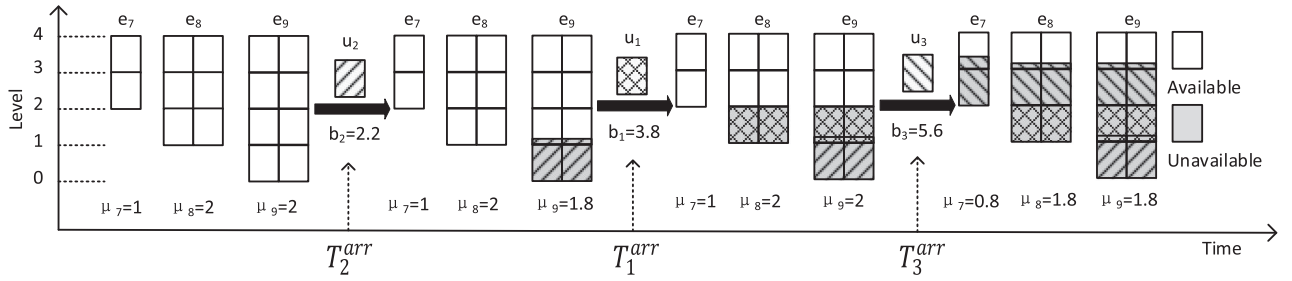
Fig. 2. An example of running LBA algorithm.

algorithm prefers allocating the computing resources of edge nodes with a small level to a newly coming task. By doing so, LBA strives to keep all edge nodes at the same level to avoid overloading edge nodes, which probably increases the probability of offloading a newly coming task to an edge node with sufficient resources. Because, otherwise, a newly coming task may find all of the edge nodes it can connect to are at the biggest level, i.e., overloaded.

Algorithm 1. LB-Allocation (LBA) alg. for ROTO-LP

1:     Initialization: $\Omega_j \leftarrow 0, \forall e_j$;
2:           $L_j \leftarrow d - V_j + \lfloor \frac{V_j \Omega_j}{B_j} \rfloor, \forall e_j$;
3:           $\mu_j \leftarrow \frac{B_j}{V_j} \cdot (L_j + 1) - \Omega_j, \forall e_j$.
4:     **if** *a new task from user u_i arrives* **then**
5:        Invoke LB shown in Algorithm 2.
6:     **if** *the task of u_i finishes* **then**
7:        **for each** *edge node e_j in Q_i* **do**
8:           $y_{ij} \leftarrow 0, \Omega_j \leftarrow \Omega_j - \omega_{ij}, L_j \leftarrow d - V_j + \lfloor \frac{V_j \Omega_j}{B_j} \rfloor$;
9:           $\mu_j \leftarrow \frac{B_j}{V_j}(L_j + 1) - \Omega_j$.

Note that, the level of each edge node is an integer, and it cannot precisely represent how full an edge node is. We introduce another important notion below.

Definition 2. *(Move-up Energy) The move-up energy μ_j of edge node e_j is the amount of computing resources e_j has to allocate to some tasks such that the level of the edge node is increased by exactly one. Formally,*

$$\mu_j \triangleq \frac{B_j}{V_j}(L_j + 1) - \Omega_j. \qquad (15)$$

Take e_9 in Fig. 2 for example, suppose $B_9 = 8$, $V_9 = 4$, and $d = 4$. Initially, $L_9 = 0$ and $\mu_9 = 2$, since it requires to allocate 2 units of resources to increase its level to 1. When the task of u_2 arrives at time T_2^{arr} and LBA allocates 2.2 units of resources to it, L_9 changes into $4 - 0 + \lfloor \frac{4 \times 2.2}{8} \rfloor = 1$, and μ_9 changes into $\frac{8}{4}(1 + 1) - 2.2 = 1.8$.

4.2.2 The Level Balanced Allocation Algorithm

The principle of LBA is as follows: *when a new task arrives, LBA gradually allocates the computing resources of the edge nodes with the smallest move-up energy among the edge nodes with the smallest level to the task.* Formally, we define the preference below.

Definition 3. *(Preference) LBA prefers allocating the computing resources of e_i than e_j if and only if*

$$L_i < L_j \qquad (16)$$

or

$$L_i = L_j \text{ and } \mu_i < \mu_j. \qquad (17)$$

If one of the two conditions holds, we denote the preference as

$$e_i \succ e_j. \qquad (18)$$

Algorithm 2. Level Balanced (LB) Procedure

1:     $b_i' \leftarrow b_i, Q_i \leftarrow \emptyset, \omega_{ij} \leftarrow 0, y_{ij} \leftarrow 0, \forall e_j \in \mathbb{E}$;
2:     **if** *each edge node e_j in $C(i)$ with $L_j = d$* **then**
3:        **break**;
4:     **else**
5:        **while** $b_i' \neq 0$ **do**
6:           $\mathbb{L} \leftarrow$ set of edge nodes with the smallest level;
7:           $h \leftarrow \arg\min_{e_j \in \mathbb{L}}(\mu_j)$.
8:           **if** $b_i' \geq \mu_h$ **then**
9:              $L_h \leftarrow L_h + 1, b_i' \leftarrow b_i' - \mu_h$;
10:            $\omega_{ih} \leftarrow \omega_{ih} + \mu_h, \Omega_h \leftarrow \Omega_h + \mu_h$;
11:            $\mu_h \leftarrow \frac{B_h}{V_h}(L_h + 1) - \Omega_h$;
12:            **if** $L_h \neq d$, **then** $Q_i \leftarrow Q_i \bigcup \{e_h\}$.
13:           **else if** $b_i' < \mu_h$ and $Q_i \neq \emptyset$ **then**
14:             **for each** *edge node e_j in Q_i* **do**
15:                $\omega_{ij} \leftarrow \omega_{ij} + \frac{b_i'}{|Q_i|}, \Omega_j \leftarrow \Omega_j + \frac{b_i'}{|Q_i|}$;
16:                $\mu_j \leftarrow \frac{B_j}{V_j}(L_j + 1) - \Omega_j, b_i' \leftarrow 0$.
17:           **else**
18:             **break**.
19:     **for each** *edge node e_j in $C(i)$* **do**
20:        $y_{ij} \leftarrow \frac{\omega_{ij}}{b_i}$.

Algorithm 1 shows the details of LBA. Lines $1 - 3$ show the initialization of Ω_j, L_j, and μ_j of each edge node. Remember that, LBA handles the online ROTO-LP problem. Lines $4 - 5$ invoke the level balanced procedure (shown in Algorithm 2) to allocate resources when a new task arrives. Lines $6 - 9$ update y_{ij}, Ω_j, L_j, and μ_j of each edge node when the task of u_i finishes.

Before we highlight the level balanced (LB) procedure, we introduce a few notations. We use ω_{ij} to denote the amount of resources allocated by e_j to the task of u_i; we use Q_i to maintain the set of edge nodes that have allocated resources to u_i; we use b_i' to represent the amount of computing demand from u_i that has not been allocated. Lines $2 - 3$ in LB check whether all edge nodes that can be reached from u_i have exhausted their computing capacities. Lines $5 - 20$ are the main loop that handles the resource allocation.

In each iteration, LB first chooses one of the edge nodes with the highest preferences, i.e., LB chooses e_h such that no other edge node e_j in $C(i)$ is more preferred by LB. If $b'_i \geq \mu_h$, then LB allocates μ_h amount of resources on e_h to the task of u_i and updates Ω_h, L_h, μ_h, ω_{ih}, and b'_i. Note that, by allocating μ_h amount of resources on e_h to the task of u_i, the level of e_h increases by exactly 1. If $b'_i < \mu_h$, the remaining computing demand of u_i cannot increase the level of any edge node by 1, LB equally splits the remaining demand into $|Q_i|$ pieces and allocates a single piece to each edge node in Q_i.

The complexity of Algorithm 1 is $\mathcal{O}(nm^2)$. Every time when a new task arrives, Algorithm 2 is invoked. LB gradually allocates the computing resources to the task. The allocation lasts at most $\mathcal{O}(m)$ rounds and each round LB checks m edge nodes. So the complexity of LB is $\mathcal{O}(m^2)$ and the complexity of Algorithm 1 is $\mathcal{O}(nm^2)$.

4.2.3 Example
Here we use a simple example to intuitively explain our algorithm. In Fig. 1, users u_1, u_2 and u_3 decide to offload their tasks to edge nodes e_7, e_8 and e_9. Assuming that e_7, e_8 and e_9 can process at most 2, 3 and 4 tasks at the same time, respectively. The computing capacity of e_7, e_8 and e_9 are 2, 6 and 8 units, respectively. The levels of e_7, e_8 and e_9 are initialized to 2, 1 and 0, respectively. μ_7 is initialized as 1 unit, and μ_8 and μ_9 are both initialized as 2 units. The running process of this example is shown in Fig. 2.

Without loss of generality, we assume that u_2 submits its task first, and its computing demand is 2.2 units. Since e_9 has the smallest level and μ_9 is smaller than b'_2 (b'_2 is initialized as 2.2 units), LBA first allocates 2 units of computing resources on e_9 to u_2 and updates the states of e_9, which makes $\mu_9 = 2$, $\Omega_9 = 2$, $\omega_{1,9} = 2$, $L_9 = 1$. Then LBA adds e_9 to Q_2. After that, b'_2 is updated to $2.2 - 2 = 0.2$. Since b'_2 is smaller than any move-up energy of edge nodes with the smallest level, LBA allocates b'_2 units of computing resources on edge nodes in Q_2, i.e., e_9. The states of e_9 are updated according to LBA, which makes $\mu_9 = 1.8$, $\Omega_9 = 2.2$ and $\omega_{1,9} = 2.2$. Then, u_1 submits its task with a demand of 3.8 units. Since e_8 and e_9 have the smallest level and μ_9 is smaller than μ_8 ($1.8 < 2$), LBA first allocates 1.8 units of computing resources on e_9 to u_1 and updates the states of e_9. In the end, 1.8 and 2 units of computing resources on e_9 and e_8 are allocated to u_1, respectively. Finally, u_3 submits its task with a demand of 5.6 units. According to LBA, e_7 allocates 1.2 units of computing resources to u_3; e_8 and e_9 each allocate 2.2 units of computing resources to u_3.

4.3 Competitive Analysis of LBA
In this section, we first formulate the dual problem of ROTO-LP and then intelligently construct the potential function, which plays an important role in the analysis. We then present two lemmas and the weak duality theorem before proving the approximation ratio of LBA.

4.3.1 Dual Problem of ROTO-LP
Here we use an indicator variable k_{ij} to help analyze the algorithm, where k_{ij} denotes whether the task of u_i is offloaded to edge node e_j. Hence

$$\forall e_j \in \mathbb{E}: \qquad \sum_{i=1}^{n} k_{ij} \leq V_j, \qquad (19)$$

$$\forall u_i \in \mathbb{U}, e_j \in \mathbb{E}: \qquad y_{ij} = k_{ij} y_{ij}, \qquad (20)$$

$$\forall u_i \in \mathbb{U}, e_j \in \mathbb{E}: \qquad k_{ij} \in \{0, 1\}. \qquad (21)$$

Put z_i to be the dual variable for user u_i, and introduce a variable x_j for edge node e_j, then the dual problem of ROTO-LP is to minimize $\sum_{j=1}^{m} B_j x_j + \sum_{i=1}^{n} z_i$ subject to

$$\forall u_i \in \mathbb{U}, e_j \in \mathbb{E}: \qquad z_i + b_i k_{ij} x_j \geq \alpha_{ij} b_i k_{ij}, \qquad (22)$$

$$\forall e_j \in \mathbb{E}: \qquad \sum_{i=1}^{n} k_{ij} \leq V_j, \qquad (23)$$

and constraints $x_j \geq 0$, $z_i \geq 0$, $k_{ij} \in \{0, 1\}$, $\forall e_j \in \mathbb{E}, u_i \in \mathbb{U}$.

Before entering the approximation analysis, we first elaborate the potential function, $f(i, j)$, which relates the values of the prime variables to that of the dual objective function.

4.3.2 Potential Function
Consider the time point when u_i just arrives, in which the ith dual constraint is given and assume that it is not satisfied. Our goal is to constrain the derivative of the dual cost (D) as a function of the primal profit (P). That is, show that $\frac{\partial D}{\partial y_{ij}} = B_j \frac{\partial x_j}{\partial y_{ij}} \leq \lambda \frac{\partial P}{\partial y_{ij}}$, where λ is going to be the competitive factor. Supposing that the derivative of the dual cost satisfies

$$B_j \frac{\partial x_j}{\partial y_{ij}} = A\left(b_i x_j + \frac{\alpha_{ij} b_i}{d}\right), \qquad (24)$$

where A is a constant. Then, since $x_j \leq \frac{\alpha_{ij} b_i}{b_i} = \alpha_{ij}$ (due to Inequality (22)), $\frac{\alpha_{ij} b_i}{d} \leq \alpha_{ij} b_i$ for $d \geq 1$, and $\frac{\partial P}{\partial y_{ij}} = \alpha_{ij} b_i$, we get that $A\left(b_i x_j + \frac{\alpha_{ij} b_i}{d}\right) \leq 2A \frac{\partial P}{\partial y_{ij}}$. Thus, $\lambda = 2A$. By solving Eq. (24), we get $\frac{\partial x_j}{\partial y_{ij}} = \frac{A}{B_j}\left(b_i x_j + \frac{\alpha_{ij} b_i}{d}\right)$. Through integration, the following equation can be obtained: $x_j = G \cdot exp\left(\frac{A}{B_j}\sum_{w=1}^{i} b_w k_{wj} y_{wj}\right) - \frac{\alpha_{ij}}{d}$, where $exp(x) = e^x$ and G can take any value. Next, we have the following two boundary conditions on this equation:

- Initially, $x_j = 0$, and this happens when $\frac{\sum_{w=1}^{i} b_w k_{wj} y_{wj}}{B_j} = 0$;

- If $\frac{\sum_{w=1}^{i} b_w k_{wj} y_{wj}}{B_j} = 1$, (i.e., the primal constraint is tight) then $x_j = \alpha_{ij}$. (Then, the dual constraint is also satisfied.)

The first boundary gives $G = \alpha_{ij}/d$. The second boundary gives $A = \ln(d + 1)$. Thus we get the potential function:

$$f(i, j) = \frac{\alpha_{ij}}{d}\left[exp\left(\frac{\ln(d+1)}{B_j}\sum_{w=1}^{i} b_w k_{wj} y_{wj}\right) - 1\right].$$

If $\sum_{w=1}^{i} b_w k_{wj} y_{wj} = B_j$, then $f(i, j) = \alpha_{ij}$. We denote $\alpha_i = \max_{e_j \in C(i)} \alpha_{ij}$. And for edge node e_j, $x_j = f(i, j)$.

4.3.3 Bounded Iteration

Let $P(i)$ and $D(i)$ be the values of the objective function of the primal and dual solutions, respectively, derived from the algorithm when u_i submits its task. Upon the arrival of a new task, we update both primal and dual programs. The primal program is updated by adding a new constraint corresponding to the user and a new term $b_i k_{ij} y_{ij}$ to each constraint of an edge node. The dual program is updated by adding a new variable z_i for the new user and a constraint of the form $b_i k_{ij} x_j + z_i \geq b_i k_{ij}$ for each edge node.

The dual solution is an assignment of values to the variables x_j and z_i. Initially, the values of primal and dual solutions are zero. Let $\Delta P(i)$ and $\Delta D(i)$ be the changes of $P(i)$ and $D(i)$, respectively, after user u_i submits its task.

Lemma 1. *In each iteration (arrival of u_i): $\Delta D(i) \leq 2(1 + \xi) \ln(d + 1) \Delta P(i)$.*

Proof. First, we consider the situation where the task of u_i is not fully allocated to edge nodes. This means that all edge nodes in $C(i)$ where $k_{ij} = 1$ have exhausted all their computing capacities after allocating computing resources to u_i. In this case, the corresponding variable x_j with $k_{ij} = 1$ are all α_{ij} at the end of the iteration. As a result, all the new dual constraints are satisfied, and we can set $z_i = 0$. For x_j with $k_{ij} = 0$, setting $z_i = 0$ also satisfies the constraints, because $b_i \cdot 0 \cdot x_j + 0 = 0 = b_i \cdot 0$. We only need to show that the change in the dual cost in this iteration is bounded. When we increase the variable y_{ij}, the derivative of the primal profit of the algorithm is $\frac{\partial(\alpha_{ij} b_i y_{ij})}{\partial y_j} = \alpha_{ij} b_i$. The derivative of the dual cost $B_j \frac{\partial f(x)}{\partial y_{ij}}$ is:

$$B_j \frac{\alpha_{ij} b_i \ln(d+1)}{d B_j} \left[exp\left(\frac{\ln(d+1)}{B_j} \sum_{w=1}^{i} b_w y_{wj} \right) \right]$$

$$= b_i \ln(d+1) \left(\frac{\alpha_{ij}}{d} \left[exp\left(\frac{\ln(d+1)}{B_j} \sum_{w=1}^{i} b_w y_{wj} \right) - 1 \right] + \frac{\alpha_{ij}}{d} \right)$$

$$= b_i \ln(d+1) \left(x_j + \frac{\alpha_{ij}}{d} \right)$$

$$\leq b_i \ln(d+1) \cdot 2\alpha_{ij} < 2(1 + \xi) \ln(d+1) \alpha_{ij} b_i.$$

The first inequality holds since $x_j = \alpha_{ij}$ and $\alpha_{ij}/d \leq \alpha_{ij}$.

Thus, we get $\Delta D(i) \leq 2(1 + \xi) \ln(d+1) \Delta P(i)$ after u_i submits its task under the condition where the task of user u_i is not fully allocated by LBA.

Then we consider the situation where the task of user u_i is fully offloaded to the edge nodes in $C(i)$. Note that the task may be offloaded to several edge nodes. We analyze this situation as follows. For edge node e_j with $k_{ij} = 0$, the derivative of the dual cost is 0, and the constraint is satisfied for all $z_i \geq 0$. For edge nodes with $k_{ij} = 1$, the derivative of the dual cost is $b_i \ln(d+1)(f(j, \alpha_{ij}) + \alpha_{ij}/d)$. We set $z_i = b_i \ln(d+1)(\alpha_i - \alpha_i/d)$ to satisfy all the new dual constraints for edge node with $k_{ij} = 1$. We can prove these constraints are satisfied as follows:

$$z_i + b_i x_j \qquad (25)$$

$$= b_i \ln(d+1) \left(\alpha_i - \frac{\alpha_i}{d} \right)$$

$$+ \frac{b_i \alpha_i}{d} \left[exp\left(\frac{\ln(d+1)}{B_j} \sum_{w=1}^{w} b_w y_{wj} \right) - 1 \right] \qquad (26)$$

$$= \frac{b_i \alpha_i}{d} \left[(d-1) \ln(d+1) + exp\left(\frac{\ln(d+1)}{B_j} \sum_{w=1}^{i} b_w y_{wj} \right) - 1 \right] \qquad (27)$$

$$\geq \frac{b_i \alpha_i}{d} \left[(d-1) \ln(d+1) + exp\left(\frac{\ln(d+1)}{d} \right) - 1 \right] \qquad (28)$$

$$\geq \frac{b_i \alpha_i}{d} \cdot d \geq \alpha_{ij} b_i. \qquad (29)$$

Inequality (28) holds since $\sum_{w=1}^{i} b_w y_{wj}/B_j \geq 1/d$ for each node with $k_{ij} = 1$. The first inequality of (29) holds since $(d-1) \ln(d+1) + exp\left(\frac{\ln(d+1)}{d} \right) - 1 \geq d$, for $d \geq 3$. Thus, all the new dual constraints are satisfied.

Therefore, when u_i arrives, the dual cost of each edge node, e_j $(z_i + B_j \Delta(x_j))$, is updated as

$$b_i \ln(d+1) \left(\alpha_i - \frac{\alpha_i}{d} \right) + b_i \ln(d+1) \left(x_j + \frac{\alpha_{ij}}{d} \right)$$

$$= b_i \ln(d+1) \left(\alpha_i - \frac{\alpha_i}{d} + x_j + \frac{\alpha_{ij}}{d} \right)$$

$$\leq b_i \ln(d+1) \left(\alpha_i - \frac{\alpha_i}{d} + x_j + \frac{\alpha_i}{d} \right)$$

$$= b_i \ln(d+1)(\alpha_i + x_j)$$

$$\leq b_i \ln(d+1) \cdot 2\alpha_i \leq 2(1 + \xi) \ln(d+1) \alpha_{ij} b_i.$$

Thus the lemma follows. $\qquad \square$

4.3.4 Feasibility

Here, we show the feasibility of LBA.

Lemma 2. *The algorithm LBA produces a feasible solution for both the primal and dual ROTO-LP problem.*

Proof. For the ROTO-LP problem, LBA never increases $\sum_{w=1}^{i} b_w y_{wj}$ to be greater than B_j. Whenever $\sum_{w=1}^{i} b_w y_{wj}$ increases in some iteration and reaches B_j, LBA stops allocating computing resources from e_j to users, because the computing capacity of e_j is exhausted. Therefore, the value of $\sum_{w=1}^{i} b_w y_{wj}$ is not going to change anymore unless some tasks processed on e_j are completed, which reduces $\sum_{w=1}^{i} b_w y_{wj}$. Also, LBA never increases $\sum_{j \in C(i)} y_{ij}$ beyond 1. Whenever $\sum_{j \in C(i)} y_{ij}$ in some iteration equals to 1, LBA stops allocating computing resources for u_i, because u_i's computing demand is satisfied.

For the dual problem of ROTO-LP, before any task on e_j is finished, the resources allocated on e_j will not decrease after the arrivals of subsequent users, which makes x_j monotonically increasing and the constraint (22) always holds. Once a task on e_j is finished, the value of x_j is either equal to 0 (no task is processed on e_j anymore) or greater than $1/d$ (at least one task is processed on e_j). We set $z_i = b_i ln(d+1)(\alpha_i - \alpha_i/d)$. If $x_j = 0$, then the

constraint (22) is satisfied for any $z_i \geq 0$. If $x_j \neq 0$, from Eqs. (25), (26), (27), (28), and (29), the constraint (22) still holds. □

4.3.5 Weak Duality

We prove weak duality of ROTO-LP here.

Theorem 2. *Let $y = (y_{11}, \ldots, y_{ij}, \ldots, y_{nm})$ and $x = (x_1, x_2, \ldots, x_m)$ be feasible solutions to the primal and dual ROTO-LP, respectively. Then:*

$$\sum_{j=1}^{m} B_j x_j + \sum_{i=1}^{n} z_i \geq \sum_{i=1}^{n} \alpha_{ij} b_i y_{ij}.$$

Theorem 2 states that the value of any feasible dual solution is at least the value of any feasible primal solution. Thus, the solution of the dual program can be used as a upper bound for any feasible primal solution. The proof of this theorem is as follows:

Proof.

$$\sum_{j=1}^{m} B_j x_j + \sum_{i=1}^{n} z_i \tag{30}$$

$$\geq \sum_{j=1}^{m} \left(\sum_{i=1}^{n} b_i k_{ij} y_{ij} \right) x_j + \sum_{i=1}^{n} z_i \tag{31}$$

$$\geq \sum_{j=1}^{m} \left(\sum_{i=1}^{n} b_i k_{ij} y_{ij} \right) x_j + \sum_{i=1}^{n} \left(z_i \sum_{j=1}^{m} y_{ij} \right) \tag{32}$$

$$= \sum_{i=1}^{n} \left(\sum_{j=1}^{m} b_i k_{ij} x_j \right) y_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{m} z_i y_{ij} \tag{33}$$

$$= \sum_{i=1}^{n} \left(\sum_{j=1}^{m} (b_i k_{ij} x_j + z_i) \right) y_{ij} \tag{34}$$

$$\geq \sum_{i=1}^{n} \alpha_{ij} b_i k_{ij} y_{ij} \tag{35}$$

$$= \sum_{i=1}^{n} \alpha_{ij} b_i y_{ij}, \tag{36}$$

where inequalities (31) and (32) hold since $\sum_{i=1}^{n} b_i k_{ij} y_{ij} \leq B_j$ and $\sum_{j=1}^{m} y_{ij} \leq 1$. Eq. (33) holds by changing the order of summation. Eq. (34) holds by merging the summation. Inequality (35) holds since $x = (x_1, x_2, \ldots, x_m)$ is feasible, which means inequality (22) is satisfied. Eq. (36) holds since $k_{ij} y_{ij} = y_{ij}$. □

Theorem 3. *Algorithm LBA is $2(1 + \xi) \ln(d + 1)$-competitive.*

Proof. Theorem 3 can be easily proved based on Lemmas 1 and 2 and the weak duality. □

4.4 The Online Algorithm for ROTO

In this subsection, we propose the solution to solve the ROTO problem, and show that the proposed algorithm achieves an approximation of $2(1 + \xi) \ln(d + 1)$ with a probability of at least $1 - e^{-\sigma n}$, where n is the number of users and σ is a constant.

4.4.1 The Level Balanced Rounding Algorithm

The pseudo code of the algorithm is shown in Algorithm 3. Upon the arrival of a user u_i, LBR invokes LB procedure and gets the fractional solution y_{ij} for offloading decisions (Line 5). The key idea of LBR is using the rounding technique to turn the fractional solution into an integer solution. The rounding steps are as follows: Since $\sum_{e_j \in \mathbb{E}} y_{ij} \leq 1$, we can map each $\sum_{w=1}^{j} y_{iw}$ (j from 1 to m) to a point with value $\sum_{w=1}^{j} y_{iw}$ on the interval [0,1], and the segment between point $\sum_{w=1}^{j-1} y_{iw}$ and $\sum_{w=1}^{j} y_{iw}$ represents the probability of offloading the task to edge node e_j. Note that $\sum_{w=1}^{m} y_{iw}$ may be smaller than 1, then the segment between point $\sum_{w=1}^{m} y_{iw}$ and point 1 represents the probability of rejecting the task of u_i.

Algorithm 3. LB-Rounding (LBR) alg. for ROTO

1: Initialization: $\Omega_j \leftarrow 0, \forall e_j$;
2:      $L_j \leftarrow d - V_j + \lfloor \frac{V_j \Omega_j}{B_j} \rfloor, \forall e_j$;
3:      $\mu_j \leftarrow \frac{B_j}{V_j} \cdot (L_j + 1) - \Omega_j, \forall e_j$.
4: **if** *a new task of user u_i arrives* **then**
5:      Invoke LB shown in Algorithm 2 for the solution y_{ij}.
6:      Choose r uniformly in the interval [0,1].
7:      **if** $r > \sum_{w=1}^{m} y_{iw}$ **then**
8:          Reject the task of u_i, $\ddot{y}_{ij} \leftarrow 0, \forall e_j \in C(i)$.
9:      **for** $j = 1$ *to* m **do**
10:          **if** $\sum_{w=1}^{j-1} y_{iw} < r \leq \sum_{w=1}^{j} y_{iw}$ **then**
11:              Allocate u_i to e_j;
12:              $\ddot{y}_{ij} \leftarrow 1, \Omega_j \leftarrow \Omega_j - \omega_{ij} + b_i, \omega_{ij} \leftarrow b_i$;
13:              $L_j \leftarrow d - V_j + \lfloor \frac{V_j \Omega_j}{B_j} \rfloor, \mu_j \leftarrow \frac{B_j}{V_j}(L_j + 1) - \Omega_j$;
14:              $\ddot{y}_{iw} \leftarrow 0, \forall w \neq j$, **break**.
15:      **for each** *edge node e_j in $C(i)$* **do**
16:          **if** $\ddot{y}_{ij} = 0$ **then**
17:              $\Omega_j \leftarrow \Omega_j - \omega_{ij}, \omega_{ij} \leftarrow 0, L_j \leftarrow d - V_j + \lfloor \frac{V_j \Omega_j}{B_j} \rfloor$;
18:              $\mu_j \leftarrow \frac{B_j}{V_j}(L_j + 1) - \Omega_j, Q_i \leftarrow Q_i \setminus \{e_j\}$.
19: **if** *the task of u_i finishes* **then**
20:      **for each** *edge node e_j in Q_i* **do**
21:          $\ddot{y}_{ij} \leftarrow 0, \Omega_j \leftarrow \Omega_j - \omega_{ij}, L_j \leftarrow d - V_j + \lfloor \frac{V_j \Omega_j}{B_j} \rfloor$;
22:          $\mu_j \leftarrow \frac{B_j}{V_j}(L_j + 1) - \Omega_j$.

We choose r uniformly in the interval [0,1] (Line 6). Then we decide whether to reject the task or to offload the task to a specific edge node according to r (Lines 7-18). We use $\ddot{y}_{ij}$ to represent the solution produced by LBR. Note that $\ddot{y}_{ij}$ is a binary variable after rounding, in general $\ddot{y}_{ij} \neq y_{ij}$. Lines 12-13 update Ω_j, L_j, and μ_j of the edge node e_j with $\ddot{y}_{ij} = 1$. Lines 15-18 update the states of each edge node e_j with $\ddot{y}_{ij} = 0$ in $C(i)$. Lines 19-22 update the states of each edge node when a task finishes.

The complexity of Algorithm 3 is $\mathcal{O}(nm^3)$. Compared to Algorithm 1, Algorithm 3 has one more step. Every time when LBA gets the fractional solution, LAR uses the rounding technique to turn the fractional solution into an integer solution. And the complexity of rounding is $\mathcal{O}(m)$. As a result, the complexity of Algorithm 3 is $\mathcal{O}(nm^3)$.

4.4.2 Analysis

Denote the objective value achieved by the LBA algorithm as Y. Denote the optimal objective value of ROTO and ROTO-LP as OPT and OPT-LP, respectively. Note that we have proved in the previous section that $Y \geq \frac{\text{OPT-LP}}{2(1+\xi)\ln(d+1)}$. Since ROTO-LP is a convex relaxation of ROTO, we have OPT-LP$\geq$ OPT. Thus, we get $2(1+\xi)\ln(d+1)Y \geq$ OPT.

Denote the objective value derived from LBR as $\ddot{Y}$. We define a random variable $\ddot{Y}_i$ such that $\ddot{Y}_i = \sum_{e_j \in C(i)} \alpha_{ij} b_i \ddot{y}_{ij}$. Then, we have $\ddot{Y} = \sum_{i=1}^{n} \ddot{Y}_i$, and $E[\ddot{Y}] = \sum_{i=1}^{n} E[\ddot{Y}_i] = \sum_{i=1}^{n} \sum_{e_j \in C(i)} \alpha_{ij} b_i \ddot{y}_{ij} = Y$.

Theorem 4. *For any $\delta \in (0, 1)$,*

$$Pr\left(\ddot{Y} < (1-\delta)\frac{\text{OPT}}{2(1+\xi)\ln(d+1)}\right) \leq e^{-\sigma n}.$$

Proof. We use the Chernoff Bound [37] to facilitate the proof.

Obviously, the random variables $\{\ddot{Y}_i\}$ are independent by construction. By applying the Chernoff Bound, we have

$$Pr\left(\ddot{Y} \leq (1-\delta)\frac{\text{OPT}}{2(1+\xi)\ln(d+1)}\right)$$
$$= Pr\left(\sum_{i=1}^{n} \ddot{Y}_i \leq (1-\delta)\frac{\text{OPT}}{2(1+\xi)\ln(d+1)}\right)$$
$$\leq Pr\left(\sum_{i=1}^{n} \ddot{Y}_i \leq (1-\delta)\frac{\text{OPT-LP}}{2(1+\xi)\ln(d+1)}\right)$$
$$\leq Pr\left(\sum_{i=1}^{n} \ddot{Y}_i \leq (1-\delta)Y\right)$$
$$= Pr(\ddot{Y} \leq (1-\delta)Y)$$
$$\leq exp(-\delta^2 Y/2).$$

Without loss of generality, we assume that OPT $= \mathcal{O}(n)$. Since $Y \geq \frac{\text{OPT-LP}}{2(1+\xi)\ln(d+1)} \geq \frac{\text{OPT}}{2(1+\xi)\ln(d+1)}$, we can find some constant $C > 0$ such that $Y \geq Cn$ for sufficiently large n. Therefore,

$$exp\left(-\frac{\delta^2 Y}{2}\right) \leq exp\left(-\frac{\delta^2 Cn}{2}\right).$$

Let $\sigma = \delta^2 C/2$, then we have

$$Pr\left(\ddot{Y} < (1-\delta)\frac{\text{OPT}}{2(1+\xi)\ln(d+1)}\right) \leq e^{-\sigma n}.$$

The theorem holds immediately. □

4.5 Discussions

In this subsection, we discuss a few limitations of our work and some possible future directions.

- *Divisible Tasks.* We assume in this work that the tasks of users are atomic. However, in many situations, a task can be divided into several independent or dependent subtasks. For dependent subtasks, we consider the subtasks that can be processed in parallel as independent tasks according to their DAG graphs. We consider independent subtasks as a new task and set the revenue of them according to their weights in the entire task.

- *Non-negligible Devices' Capacities.* Our work focus on the scenarios that the mobile device has very limited computing resource, and hence all the tasks should be offloaded to the edge nodes for execution [38]. Under the condition where the mobile device has more powerful computing capacity and the task is divisible, we can consider the user as a special edge node, which only connects to the user itself, and set the revenue as the energy consumption if the task is processed locally. LBA can be slightly adjusted to suit this condition.

- *Dynamic Revenue.* Our work assumes that the revenue of the task is fixed. However, users may be willing to pay more for their tasks due to the urgency. The revenue ratio α_{ij} can be dynamically modified according to current time and deadline, which means scheduling some urgent tasks can bring in more revenue. Meanwhile, because of the existence of competition, users need to consider how to modify their prices so that their tasks can be completed at the edge node with minimal cost. We plan to use game-theoretic framework to study this problem.

- *Uncertain Task Requirements.* We mainly talk about the scenario where the computing requirement of each task is available. However, in realistic scenarios, the size of the task can be measured but its processing time is generally uncertain until it is completed. Task assignment under the uncertainty of the processing time is well studied in theoretical computer science [39], [40]. However, most works focus on the design of efficient task scheduling and do not concern the allocation of computing resources. We would like to study the computation offloading problem under uncertain processing time in the future.

- *Nonnegligible Transmission Time.* In this paper, we assume that with the rapid development of $5G$ network, the time of transmission is small enough to be ignored [33]. Meanwhile, we assume that users arrive one by one and the task of the user can only be offloaded to the edge nodes that are within the communication range of the same BS as the user. As a result, the scale of the ROTO problem will not be particularly large. However, for large-scale MEC scenes, the performance of the algorithm may not be as good as theoretically. Collecting and scheduling a large amount of edge nodes in a short period of time is a big challenge. We plan to use a distributed competition method to analysis the problem when considering the transmission delay. Specifically, we plan to design a self-organizing distributed framework, that is, whenever a user arrives, the scheduling calculations of task offloading is only performed in the BS to which the user can connect.

5 TRACE-DRIVEN SIMULATION

In this section, we demonstrate the performance of LBR. We compare LBR with three baseline algorithms. Both of them
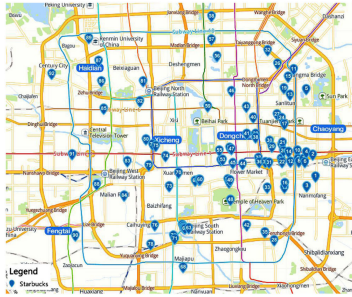
Fig. 3. Loc. of starbucks.

Fig. 4. Our testbed.

assume that the information of all users is known in advance. The first one is the Random Allocation (RA) algorithm, when the user arrives, RA randomly offloads the task to an edge node connected to the user. The second one is the Greedy Allocation (GA) algorithm, which always offloads tasks to the edge node with the maximum revenue. The third one is the optimal (OPT) solution, which is obtained from the existing mathematical tools (IBM CPLEX [41]). After presenting the setup and parameters, the results are shown from different perspectives to provide insightful conclusions.

5.1 Simulation Settings

We envision a mobile edge computing system deployed inside a megacity in Asia.

Edge Node Trace. For the locations of envisioned edge nodes, we use the locations of the Starbucks due to the fact that Starbucks shops in a city can usually achieve a decent coverage for users. In addition, the distribution of Starbucks actually follows the population density, making them perfect locations for edge cloud deployment in the future. We collect the locations of 105 Starbucks in the megacity, which is shown in Fig. 3. Each Starbucks is envisioned as an edge node. Then the default number of edge nodes is 105. The computing capacity of an edge node is in a range of $3 - 4$ GHz and the process slots of each edge node is in a range of $4 - 12$.

Task Trace. The task statistics are in accordance with [31]. For tasks, the expected input data size per task is 8 MB, and the expected number of CPU cycles required to process one byte of data is 1000, which makes the computing demand of users be in a range of 0.27-0.4 gigacycles. The price statistics are in accordance with [35], [36]. The value of the price of unit computing demand α_{ij} is set between 0.5 and 0.6.

User Trace. The number of customers visiting a Starbucks shop in a day in the megacity is about 500, which implies
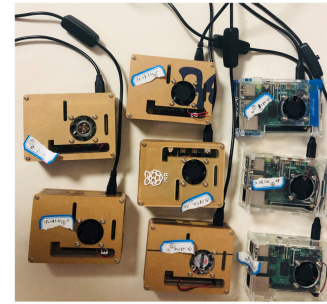
about 50 customers in an hour. We set the total running time as $|\mathbb{T}| = 100$. We set the default number of users to 5000. Locations of users are randomly distributed in the megacity. In the 5G scenario, we consider a user can connect to an edge node if the distance between them is within the communication range, and the default communication range is 3km. For users, we use the data traces of Google clusters [42]. Note that, these traces only contain the information of processing time, dependency relationship and required processing resources for each task. We adjust users' arrival time based on Poisson distribution to accommodate stochastic arrival and the length between deadlines and arrival time are uniformly distributed in $[20, 30]$.

5.2 Simulation Results

In Fig. 5a, we evaluate the influence of the number of users which varies from 2000 to 6000. We observe the trend that as the number of users grows, the overall revenue grows accordingly. This is reasonable because more users will be served to maximize the revenue as long as the edge nodes have sufficient computing capacity. We also observe that, the overall revenue does not exceed a certain threshold. This is because the computing capacity of edge nodes is limited. Once edge nodes exhaust their capacities, the future users will not be served, thus the revenue will no longer increase. Our online algorithm always outperforms the RA and GA and is close to the optimal algorithm.

Fig. 5b shows the impact of the number of edge nodes on overall revenue. We randomly choose $60 - 100$ edge nodes among 105 edge nodes. We observe that the overall revenue increases as the number of edge nodes increases. This is because, with the increase of the number of edge nodes, the computing capacity has increased accordingly. And more computing capacity will also bring more revenue. Still, our
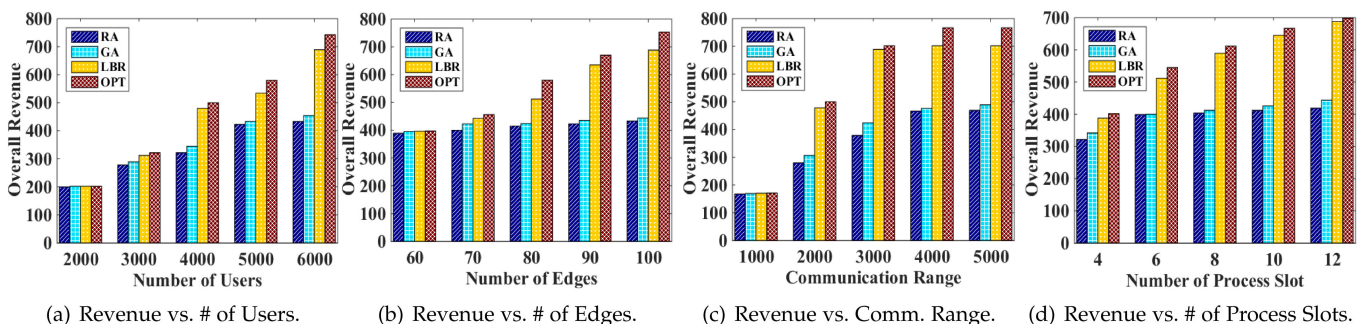

(a) Revenue vs. # of Users.

(b) Revenue vs. # of Edges.

(c) Revenue vs. Comm. Range.

(d) Revenue vs. # of Process Slots.

Fig. 5. Simulation results for our algorithm and the baseline algorithms with different configurations.

(a) Revenue vs. Running Time.   (b) Revenue vs. # of Users.   (c) Revenue vs. # of Edges.   (d) Revenue vs. # of Connections.
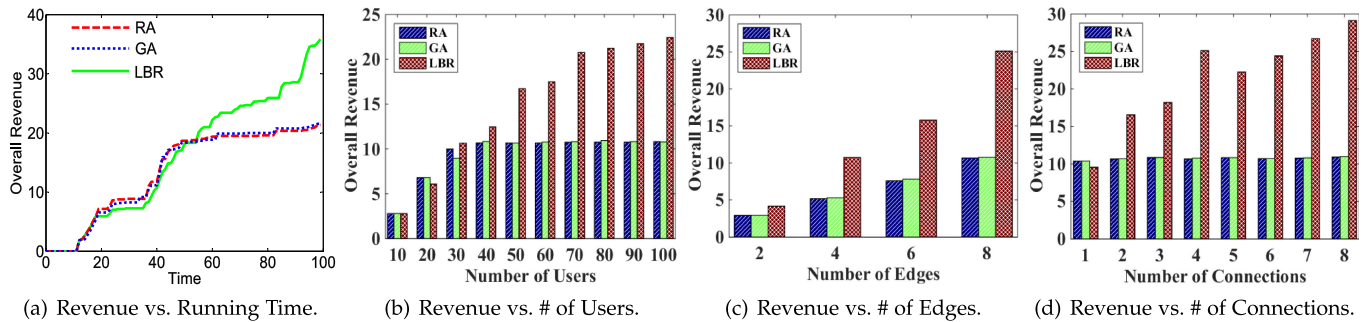
Fig. 6. Testbed results for our algorithm and the baseline algorithms with different configurations.

algorithm performs better than RA and GA and is close to the optimal algorithm.

In Fig. 5c, we inspect the impact of the network topology. We vary the communication range from 1km to 5km, which makes the average number of edge nodes that connect to a user vary between 1 and 20. We observe that, the overall revenue grows as the number of connections increases. This is as expected, because when there are few connections, it is very likely that all the edge nodes that connected to the user have exhausted their computing capacities, while other edge nodes still have sufficient capacities. With more connections, this happens less likely. And the overall revenue in our algorithm is significantly higher than that in RA and GA and is close to the optimal algorithm.

Fig. 5d shows the results with different number of process slots. We vary the number of process slots of each edge node from 4 to 12. We can see from Fig. 5d, as the number of process slots grows, the overall revenue grows. This is because, with more process slots more users can be served by each edge node, thus the overall revenue increases. And our algorithm outperforms the classic RA and GA algorithms and is close to the optimal algorithm.

The reasons why LBR outperforms GA and RA will be discussed in Section 6.2.

6 TESTBED EXPERIMENT

In this section, we implement the revenue-driven online task offloading system and conduct experiments based on the implementation to validate the performance of LBR.

6.1 Testbed Setting

Deployment Platforms. The testbed environment is shown in Fig. 4; we use 5 Raspberry Pi 4 Model B (ARM Cortex-A72 CPU, 4 Cores @ 1.5 GHz)) and 3 Raspberry Pi 3 Model B (ARM Cortex-A53 CPU, 4 Cores @ 1.2 GHz) as edge nodes. We use 2 Samsung S4 and 2 Samsung note3 phones as users, and each user is connected to $3 - 5$ Raspberry Pis. The program is written in python3.7 environment, and has roughly implemented with about 2.3k lines of codes.

User Trace. We adjust users' arrival time based on Poisson distribution. The time between the arrival time and the deadline of a task are uniformly distributed in [10,15].

Task Trace. Consider the realistic task trace from Google cluster as computation-intensive tasks, which contains the information of processing time, dependency relationship and required processing resources for each task [42]. The

expected number of CPU cycles required to process one byte of data is in accordance with [31]. We set up two different tasks in our experiments, namely pdf2text and html2text programs. The expected number of CPU cycles required to process one byte of data of pdf2text and html2text is 1000 and 6000, respectively. The value of the price of unit computing demand α_{ij} is set between 0.5 and 0.6. We set the size of each pdf2text task between 1.05 MB and 4.45 MB, and the size of each html2text task is set between 5.9 MB and 9 MB. We set the total running time as $|\mathbb{T}| = 100$.

6.2 Performance Comparison

Fig. 6a shows the change in overall revenue over time. Note that an edge node can only gain the revenue after the task is successfully executed. In the first 10 seconds, the total revenue remains at 0 because no tasks have been completed yet. We observe that the performance of all algorithms is very similar in the beginning. This is because the computing resources of edge nodes were sufficient in the beginning and all tasks can be offloaded to edge nodes. We also observe that GA and RA perform better than LBR in the beginning, this is because LBR always selects the edge node with the smallest level, which may not bring the biggest revenue. However, LBR outperforms baselines over time, and the overall revenue in LBR is 50 percent higher than that in the baselines in the end.

The experimental results on the testbed are consistent with our simulation results. Figs. 6b and 6c show the overall revenue versus the number of users and the number of edge nodes, respectively. The overall revenue increases with the growth of the number of users and the number of edge nodes, and LBR achieves significant revenue increment compared to other algorithms. Fig. 6d evaluates the impact of the number of connections between users and edge nodes on the overall revenue. By increasing the number of connections, the overall revenue will also increase.

The reason why LBR outperforms GA and RA in Figs. 5a, 5b, 6b and 6c is as follows: In the beginning, GA prefers to offload tasks on the edge nodes with the maximum revenue, which causes imbalance in the remaining resources of edge nodes. Thus, some edge nodes may exhaust their resources quickly while others still have sufficient resources. As a result, the newly arrived task is very likely to be rejected because all of the edge nodes it can connect to are overloaded. This situation may also occur when applying RA.

The reason why LBR outperforms GA and RA in Figs. 5c, 5d, and 5d is as follows. The number of connections and process slots will not affect the offloading decisions of GA and RA. More connections and process slots will only slow down the speed at which some edge nodes exhaust their resources, and cannot change the result that some edge nodes exhaust their resources while others have sufficient resources.

7 CONCLUSION

In this paper, we formulate the task computation offloading as an optimization problem to maximize offloading revenue while providing performance guarantees. The proposed algorithm achieves the approximation of $2(1 + \xi)\ln(d + 1)$ with a probability of at least $1 - e^{-\sigma n}$. Trace-driven simulations and testbed experiments have shown that our proposed scheme outperforms the classic RA and GA algorithms and is close to the optimal algorithm.
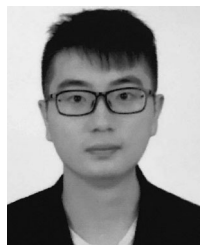
REFERENCES

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

[2] Z. Zhang and W. Hao, "Development of a new cloudlet content caching algorithm based on web mining," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf.*, 2018, pp. 329–335.

[3] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Fourth Quarter 2017.

[4] C. Li, J. Tang, and Y. Luo, "Dynamic multi-user computation offloading for wireless powered mobile edge computing," *J. Netw. Comput. Appl.*, vol. 131, pp. 1–15, Apr. 2019.

[5] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 726–738, Sep./Oct. 2018.

[6] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, 2019, Art. no. 1446.

[7] S. Yu, X. Chen, L. Yang, D. Wu, M. Bennis, and J. Zhang, "Intelligent edge: Leveraging deep imitation learning for mobile edge computation offloading," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 92–99, Feb. 2020.

[8] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[9] N. Maurice, Q.-V. Pham, and W.-J. Hwang, "Online computation offloading in noma-based multi-access edge computing: A deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 99 098–99 109, 2020.

[10] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.

[11] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[12] G. Qiao, S. Leng, and Y. Zhang, "Online learning and optimization for computation offloading in D2D edge computing and networks," *Mobile Netw. Appl.*, pp. 1–12, 2019.

[13] A. Samanta and Z. Chang, "Adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint," *IEEE Internet of Things J.*, vol. 6, no. 2, pp. 3864–3872, Apr. 2019.

[14] J. Huang, S. Li, and Y. Chen, "Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing," *Peer-to-Peer Netw. Appl.*, vol. 13, pp. 1776–1787, 2020.

[15] D. Zhang *et al.*, "Near-optimal and truthful online auction for computation offloading in green edge-computing systems," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 880–893, Apr. 2020.

[16] Z. Cao, H. Zhang, B. Liu, and B. Sheng, "A game-theoretic framework for revenue sharing in edge-cloud computing system," in *Proc. IEEE 37th Int. Perform. Comput. Commun. Conf.*, 2018, pp. 1–8.

[17] "Aliyun edge," 2021. [Online]. Available: https://blog.karatos.in/a?ID=01050-d5c12e84-ca86-458f-abb7-adf595688385

[18] "Amazon edge," 2021. [Online]. Available: https://aws.amazon.com/lambda/edge/

[19] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

[20] L. Liu, H. Tan, S. H.-C. Jiang, Z. Han, X.-Y. Li, and H. Huang, "Dependent task placement and scheduling with function configuration in edge computing," in *Proc. IEEE/ACM 27th Int. Symp. Qual. Service*, 2019, pp. 1–10.

[21] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[22] T. Zhu, J. Li, Z. Cai, Y. Li, and H. Gao, "Computation scheduling for wireless powered mobile edge computing networks," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1–9.

[23] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading dependent tasks in mobile edge computing with service caching," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1–9.

[24] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1–9.

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

[26] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1468–1476.

[27] S. Jošilo and G. Dán, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2467–2475.

[28] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Internet of Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[29] H. Guo, J. Liu, and J. Zhang, "Efficient computation offloading for multi-access edge computing in 5G hetnets," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.

[30] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[31] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, Art. no. 19.

[32] H. Yuan, J. Bi, W. Tan, and B. H. Li, "Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 337–348, Jan. 2017.

[33] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5G ultra-dense cellular networks," *IEEE Wireless Commun.*, vol. 23, no. 1, pp. 72–79, Feb. 2016.

[34] X. Tao, H. Qi, W. Li, K. Li, and Y. Liu, "Profit-aware scheduling in task-level for datacenter networks," *Comput. Elect. Eng.*, vol. 61, pp. 327–338, Jul. 2017.
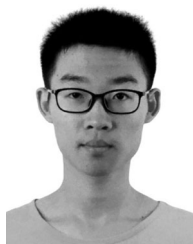
[35] R. Li, Z. Zhou, X. Chen, and Q. Ling, "Resource price-aware off-loading for edge-cloud collaboration: A two-timescale online control approach," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2937928.

[36] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 420–423, Jul. 2018.

[37] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[38] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 760–770, Third Quarter 2016.

[39] N. Eshraghi and B. Liang, "Joint offloading decision and resource allocation with uncertain task computing requirement," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1414–1422.

[40] M. Hu, L. Zhuang, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Learning driven computation offloading for asymmetrically informed edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1802–1815, Aug. 2019.

[41] S. Nickel, C. Steinhardt, H. Schlenker, W. Burkart, and M. Reuter-Oppermann , "IBM ILOG CPLEX optimization studio," in *Angewandte Optimierung mit IBM ILOG CPLEX Optimization Studio*. Berlin, Germany: Springer, 2021, pp. 9–23.

[42] Google, "Google cluster," 2020. [Online]. Available: https://code.google.com/p/googleclusterdata/

Zhiqi Chen is currently working toward the master's degree at the Department of Computer Science and Technology, Nanjing University, China. His research interests include NFV placement, mobile edge computing, and optimization.
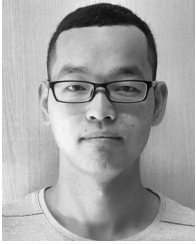
Zhi Ma received the BS degree from Nanjing University, China, in 2017, where he is currently working toward the PhD degree under the supervision of Prof. Sheng Zhang. His research interests include wireles charging and edge computing.

Sheng Zhang (Member, IEEE) received the BS and PhD degrees from Nanjing University, in 2008 and 2014, respectively. He is currently an associate professor with the Department of Computer Science and Technology, Nanjing University. He is also a member of the State Key Laboratory for Novel Software Technology. His research interests include cloud computing and edge computing. To date, he has published more than 80 papers, including those appearing in *IEEE Transactions on Mobile Computing*, *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, MobiHoc, ICDCS, INFO-COM, SECON, IWQoS, and ICPP. He received the Best Paper Award of IEEE ICCCN 2020 and the Best Paper Runner-Up Award of IEEE MASS 2012. He was the recipient of 2015 ACM China Doctoral Dissertation Nomination Award. He is a senior member of the CCF.

Tao Han (Member, IEEE) is currently an associate professor in the Department of Electrical and Computer Engineering at New Jersey Institute of Technology. He received the PhD degree in electrical engineering from New Jersey Institute of Technology (NJIT) in 2015. He was the recipient of IEEE International Conference on Communications (ICC) Best Paper Award 2019, IEEE Communications Society's Transmission, Access, and Optical Systems (TAOS) Best Paper Award 2019, Newark College of Engineering Outstanding Dissertation Award 2016, NJIT Hashimoto Prize 2015, and New Jersey Inventors Hall of Fame Graduate Student Award 2014. He serves as an associate editor of IEEE Communications Letters and TPC member for numerous IEEE conferences. His research interest includes mobile edge networking, machine learning, mobile X reality, 5G system, Internet of Things, and smart grid.

Zhuzhong Qian (Member, IEEE) received the PhD degree in computer science, in 2007. He is a professor with the Department of Computer Science and Technology, Nanjing University, P. R. China. His research interests include cloud computing, distributed systems, and pervasive computing. He is the chief member of several national research projects on cloud computing and edge computing.

Mingjun Xiao (Member, IEEE) received the PhD degree from the University of Science and Technology of China, in 2004. He is currently a professor with the School of Computer Science and Technology, University of Science and Technology of China (USTC). His research interests include mobile crowdsensing, blockchain, edge computing, mobile social networks, vehicular ad hoc networks, auction theory, data security and privacy. He has published more than 90 papers in referred journals and conferences, including the *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Networking*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Services Computing*, INFOCOM, ICDE, ICNP, ICDCS, etc. He served as the TPC member of many top conferences, including INFO-COM18-21, IJCAI 21, DASFAA20, ICDCS19, etc. He is on the reviewer board of several top journals such as *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Networking*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Cloud Computing*, etc.

Ning Chen received the BS degree from the Chongqing University of Post and Telecommunication, in 2018. He is currently working toward the PhD degree at the Department of Computer Science and Technology, Nanjing University, under the supervision of Prof. Sheng Zhang. His research interests including edge computing, deep reinforcement learning, and video streaming. His publications include those appearing in the *IEEE Transactions on Parallel and Distributed Systems*, SECON, ICPADS and Elsevier Computer Network.

Sanglu Lu (Member, IEEE) received the BS, MS, and PhD degrees in computer science from Nanjing University, in 1992, 1995, and 1997, respectively. She is currently a professor with the Department of Computer Science and Technology, State Key Laboratory for Novel Software Technology. Her research interests include distributed computing, wireless networks, and pervasive computing. She has published more than 80 papers in referred journals and conferences in the above areas.

Jie Wu (Fellow, IEEE) is currently the director of the Center for Networked Computing and the Laura H. Carnell professor with Temple University. He also serves as the director of International Affairs at College of Science and Technology. He served as chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and associate vice provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director with the National Science Foundation and was a distinguished professor with Florida Atlantic University. His research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including the *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Service Computing*, *Journal of Parallel and Distributed Computing*, and *Journal of Computer Science and Technology*. He was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society distinguished visitor, ACM Distinguished Speaker, and chair for *IEEE Technical Committee on Distributed Processing* (TCDP). He is a CCF Distinguished speaker. He was the recipient of 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

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