Distributed Offloading in Overlapping Areas of Mobile-Edge Computing for Internet of Things

Jiwei Huang, *Member, IEEE*, Ming Wang, Yuan Wu, *Senior Member, IEEE*, Ying Chen, *Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—With the maturity of 5G cellular communication systems and mobile-edge computing (MEC), a large number of base stations (BSs) with edge-computing servers are densely deployed. There are extensive overlapping coverage areas among the BSs in which some heavy computational tasks from Internet of Things (IoT) devices can be divided and offloaded to multiple BSs via the coordinated multipoint (CoMP) technique for parallel processing. However, it is a challenging issue about how to make proper task offloading decisions among multiple connected BSs while satisfying delay requirements of multiple devices. To address this challenge, this article presents an efficient multi-device and multi-BSs task offloading scheme with the goal of minimizing the delay for completing the tasks of the devices. By conducting quantitative analysis of local delay and offloading delay, a nonlinear and nonconvex delay optimization offloading problem, which is based on the theory of noncooperative game, is formulated. We prove the existence of Nash equilibrium by analyzing the feature of the proposed offloading problem and further propose a distributed task offloading algorithm called DOLA. Finally, simulation experiments based on real-world data set from the Melbourne CBD area of Australia are conducted to validate the efficacy of our DOLA algorithm. Comparison experiments are also carried out to demonstrate the superiority of DOLA in comparison with some existing schemes.**

***Index Terms*—Distributed task offloading, Internet of Things (IoT), mobile-edge computing (MEC), Nash equilibrium.**

I. Introduction

IN RECENT years, there has been an explosive growth in the number of Internet of Things (IoT) devices, and a variety of delay-sensitive applications (e.g., real-time sensing, video processing, and dynamic control) has been deployed on the IoT devices [1], [2]. These complex applications on the IoT devices generate a large amount of data. It is estimated in [3] that the mobile data traffic produced by these applications and services would reach 49 EB in global terms. However, the computing resources and the battery capacities of the IoT devices are limited. Therefore, it is generally difficult and even impossible for these IoT devices to process all the application tasks locally while satisfying the performance demands of applications.

To address this issue, computation offloading is considered a promising solution [4]. IoT devices can offload their computation-intensive tasks to remote cloud data centers for processing [5], [6]. The cloud data centers have abundant computing resources and unlimited power supply. However, computation offloading to cloud data centers also faces several challenges. As the data traffic generated by the IoT devices is increasing dramatically, offloading all the data to the remote cloud would put heavy burden on the network and cause network congestion. Besides, the cloud data centers are usually located far away from terminal devices. Therefore, offloading tasks to the remote cloud would experience large delay and poor performance. Fortunately, with the development of communication technology, especially the 5th generation mobile communication technology (5G) [7], [8], mobile-edge computing (MEC), which deploys the intensive computation resources close to the edge of wireless networks [e.g., the cellular base stations (BSs) or the local wireless access points], has been envisioned as a promising paradigm to address the aforementioned shortcoming of the conventional cloud systems and eventually enable the computation-intensive and latency-sensitive mobile Internet services [1], [9]. In MEC, a BS can be equipped with one or multiple edge servers to provide the computing services for the IoT devices, which thus effectively reduce the communication overhead and the delay [3], [10]. The advantage of MEC has also attracted lots of interests in exploiting MEC for various IoT services [11].

Nowadays, the 5G BSs are deployed more densely [12], and an IoT device may be located in the overlapping coverage areas of several BSs. Thus, an IoT device can be connected to multiple BSs, and can offload its tasks to multiple BSs

Manuscript received 2 August 2021; revised 23 November 2021; accepted 7 January 2022. Date of publication 18 January 2022; date of current version 25 July 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61972414, Grant 61902029, Grant 62072490, and Grant 61973161; in part by the Beijing Nova Program under Grant Z201100006820082; in part by the Beijing Natural Science Foundation under Grant 4202066; in part by the Excellent Talents Projects of Beijing under Grant 9111923401; in part by the Scientific Research Project of Beijing Municipal Education Commission under Grant KM202011232015; in part by the FDCT-MOST Joint Fund Project under Grant 0066/2019/AMJ; and in part by the Macao Science and Technology Development Fund under Grant 0060/2019/A1 and Grant 0162/2019/A3. *(Corresponding author: Ying Chen.)*

Jiwei Huang and Ming Wang are with the Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China (e-mail: huangjw@cup.edu.cn; 2019211263@student.cup.edu.cn).

Yuan Wu is with the State Key Laboratory of Internet of Things for Smart City, University of Macau, Macau, China, and also with the Zhuhai-UM Science and Technology Research Institute, Zhuhai 519000, China (e-mail: yuanwu@um.edu.mo).

Ying Chen is with the Computer School, Beijing Information Science and Technology University, Beijing 100101, China (e-mail: chenying@bistu.edu.cn).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/JIOT.2022.3143539

for processing. Computation offloading to the edge servers on multiple BSs can make full use of the resources of these edge servers and further reduce the delay, and the Quality of Experience (QoE) of the devices can be improved greatly [13]. However, with far more increase in the number of IoT devices and the data traffic generated by these devices' applications, the communication resources and the computing capacities of these BSs may be still limited and face shortages. If the task offloading decisions are made improperly, the workload among these edge servers on BSs may be imbalanced, and some edge servers may be heavy loaded while others suffer from a low utilization, resulting in large delay and even task congestion leading to a system crash. Besides, multiple IoT devices compete for the limited communication and computing resources, and it is a challenging issue for network operators or service providers to make a global decision of offloading strategy obtaining equilibrium. Therefore, how to find a proper task offloading scheme among multiple connected BSs while satisfying the requirements of multiple devices remains largely unexplored.

This article studies distributed task offloading in overlapping areas of MEC for IoT. The tasks from the device can be separated into multiple subtasks, and executed locally or offloaded to multiple BSs in parallel. We theoretically analyze the property of the offloading problem, formulate the global offloading game model, and propose the distributed task offloading algorithm (DOLA) for all the IoT devices. Then, we adopt real-world data from Melbourne CBD area of Australia to evaluate the performance of our DOLA algorithm.

The main contributions of our work are as follows.

1) We consider the scenario of a MEC system with multiple IoT devices and densely deployed BSs. Each IoT device is situated in the overlapping area of different BSs, and the task from the IoT device can be divided into several subtasks. We formulate the offloading optimization problem for each device, and the goal is to minimize the delay subject to resource constraints.
2) We theoretically analyze the property of the delay optimization problem for each IoT device. Then, the offloading delay optimization problem is formulated as a global offloading game model. We theoretically prove the existence of Nash equilibrium, and propose the DOLA algorithm for all devices to converge to the Nash equilibrium.
3) In order to verify the performance of our DOLA algorithm, we carry out the experiments with real-world data set from Melbourne CBD area of Australia. The experimental results show that our DOLA algorithm converges quickly, and can achieve good scalability and work-balance performance when the system scale increases.

The remainder of this article is organized as follows. Section II reviews the related work. Section III presents the system model and delay optimization problem formulation. Section IV analyzes the property of the offloading problem and proposes the DOLA algorithm. Section V evaluates the performance of our DOLA algorithm with real-world trace data, and Section VI concludes this article and discusses the future direction.

II. Related Work

In this part, we discuss related work on task offloading from two aspects: 1) coarse-grained offloading, where a task cannot be further divided and 2) fine-grained partial offloading, where a task can be divided into several subtasks.

A. Coarse-Grained Offloading

Coarse-grained offloading means that a task from the device is nondivisible, and the whole task is either processed locally or offloaded (to MEC servers or cloud servers). In [14], a MEC system with a single-user and multiple tasks was investigated. In order to jointly optimize the computation time and the energy consumption of all tasks, they formulated a minimization problem as a nonconvex mixed-integer problem. Furthermore, Wu *et al.* [15] studied a MEC system with a BS and several mobile terminals. Mobile terminals offload partial of their tasks to a BS that is equipped with an edge server, and the authors aimed to minimize the execution time of all mobile terminals. Chen *et al.* [16] focused on a multilevel offloading scenario that included a user with multiple tasks, an access point with computation resource, and a cloud server. There were three options for the user to choose to process the task, i.e., locally, on the MEC server and on the cloud server. With the same optimal target in [14], the goal was to optimize the overall cost of energy and delay, and they designed a semidefinite relaxation and randomization mapping algorithm to make the offloading decision.

Some works focused on multiple users offloading scenarios. Chen [17] aimed to minimize the cost defined by a weighted sum of latency and power. They proposed an efficient offloading strategy based on game theory, and a Nash equilibrium of the game was obtained. Unfortunately, the authors only developed the model for one single access point. In order to study a more complex system, Jošilo and Dán [18] considered a multiusers and multiaccess points scenario, and each user could select one wireless access point to transmit data to cloud for computation offloading. Furthermore, Zhang *et al.* [19] took fairness among users during task offloading into consideration, and they formulated a problem to minimize the maximal task execution time among all the users. Guo and Liu [20] studied cloud-MEC collaborative task offloading, and made use of the computation resource of the cloud. Zhou *et al.* [21] aimed to maximize the expected offloading rate of multiple agents by optimizing their offloading thresholds.

B. Fine-Grained Partial Offloading

Fine-grained partial offloading means the task of the device can be divided into multiple subtasks, and the subtasks can be processed locally, or on offloading servers in parallel. Generally speaking, partial offloading is more complex compared with coarse-grained offloading as the solution space size is much larger. Cao *et al.* [22] considered a scenario, including a single user node, a help node, and a MEC server. The user could divide the task into three parts that were processed locally, on help node and MEC server. The goal was to optimize the energy consumption of the user under the execution latency constraint. Furthermore, in order to solve the

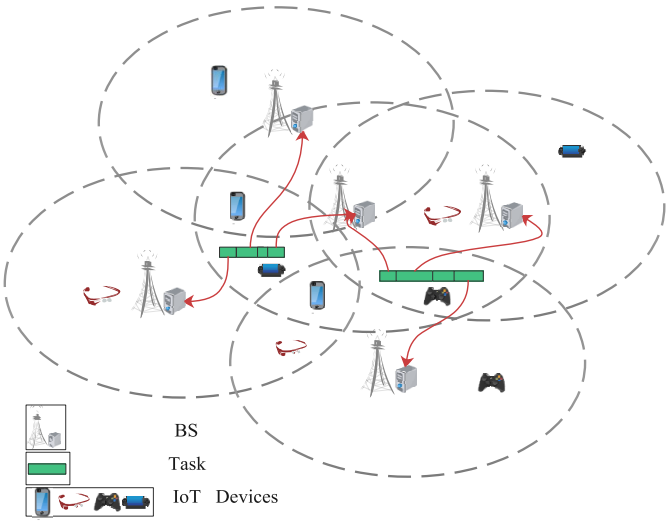

Fig. 1. Partial computation offloading scenario in overlapping area.

multiuser with a single edge server computation offloading problem, Le *et al.* [23] proposed a method to minimize the computation time for each user by joint allocation of radio and computation resource. In order to utilize the computation resource of cloud, Mahn *et al.* [24] considered a MEC system with the multilevel architecture, in which there were multiple devices, an access point with computation resource, and a remote cloud server. They aimed to minimize the energy consumption of each user, and a distributed algorithm was designed to obtain the fractions of subtasks.

Some works considered the multi-TN (task node) and multi-HN (help node) scenario in a fog network [25], [26]. In [25], each TN had a splittable task that could be divided into multiple subtasks, and those subtasks could only be offloaded to multiple HNs and virtual machines on the cloud. The authors defined a concept named processing efficiency, and a decentralized task scheduling algorithm was designed to minimize the computation delay. However, they assumed each HN and virtual machine could process only one subtask. Then, in [26], they improved their work, and the HNs could execute multiple subtasks from different TNs.

In this article, we study the partial offloading scenario where a number of devices are situated in overlapping areas of multiple BSs. The tasks of the device can be divided into several subtasks and be executed locally and on different BSs in parallel. We theoretically analyze the property of the offloading problem, and then, prove the existence of Nash equilibrium and propose the DOLAs for all the devices.

III. System Model and Problem Formulation

A. Scenario

We consider a cellular network shown in Fig. 1, and it consists of N IoT devices denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$ and K BSs denoted by $\mathcal{K} = \{1, 2, \ldots, K\}$. These devices are situated at the overlapping coverage of multiple BSs. Each BS $k \in \mathcal{K}$ is endowed with a MEC server, which provides computing services to the devices within its radio range. The coordinated multipoint (CoMP) technique [27] is adopted to ensure that a device sends data to multiple BSs for processing, and the result can be returned back through one BS. The tasks are divided into multiple subtasks and offloaded to multiple BSs for parallel computation. CoMP has been considered as a promising technique for enabling cooperative transmission among multiple BSs or transmission points (TPs) [28]. Via CoMP, the co-channel interference among different devices can be effectively mitigated by sharing the channel state information (CSI) among different BSs. With CoMP, the network efficiency and throughput can be improved in network coordination. Our main focuses are to study the optimal computation offloading under the scenario of CoMP for task processing and the corresponding algorithmic design for achieving the optimal solution.

In this article, we are concerned with arbitrarily partitioned tasks, i.e., tasks that can be divided into infinite granularity [29]. x_n^k denotes the percentage of subtask offloaded from device n to BS k, and $x_n^k \in [0, 1]$, $k \in \{0\} \cup \mathcal{K}$. x_n^0 is the fraction of task processed locally, and it is obvious that $x_n^0 + \sum_{k \in \mathcal{K}} x_n^k = 1$. Thus, the offloading strategy matrix is $\mathbf{X} = (\mathbf{x}_1^\mathsf{T}, \mathbf{x}_2^\mathsf{T}, \ldots, \mathbf{x}_N^\mathsf{T})^\mathsf{T}$, where $\mathbf{x}_n = (x_n^0, x_n^1, \ldots, x_n^K)$ is the offloading policy for device n. We introduce a connection matrix $\mathbf{C} = (\mathbf{c}_1^\mathsf{T}, \mathbf{c}_2^\mathsf{T}, \ldots, \mathbf{c}_N^\mathsf{T})^\mathsf{T}$ representing the connection between the devices and the BSs. $c_n^k = 1$ means device n is connected to BS k; otherwise, $c_n^k = 0$ indicates there is no connection between device n and BS k. The main notations in this article are listed in Table I.

B. Local Computation and Offloading Computation

Here, we build the computation models, including local computation and offloading computation. Similar to many related works in the research area of parallel offloading, we consider the scenario where tasks can be partitioned into multiple independent parts, such as face detection, virus scan, and AR applications. Device n's task can be divided into multiple subtasks, and the subtasks will be computed parallelly both locally and on multiple BSs. The computation delay of local computation is

$$T_n^{0,\mathrm{co}} = x_n^0 \beta_n \frac{l_n}{f_n} \tag{1}$$

where l_n is the size of the task (in bits), β_n represents the processing density, i.e., the number of CPU cycles required to process one bit data, and f_n is the CPU frequency of the device n.

Each device transmits the subtasks to the connected BSs via wireless channel. Several BSs will process the subtasks in parallel, and return the result to the device through one BS. Similar to most of the previous works [29], [30], since the size of the result is very small compared with the original data size, we ignore the time for sending the result back to the device. We consider that the BSs serve devices via frequency-division multiple access (FDMA) schemes [24] and the channel is preallocated. Therefore, there is no interference between different devices.

Similar to existing works [26], we consider all subtasks on the BSs must be transmitted before starting computing in

TABLE I
NOTATIONS AND DEFINITIONS

Notations	Definitions
$\mathcal{N}$	set of devices
$\mathcal{K}$	set of BSs
N	number of devices
K	number of BSs
x_n^k	percentage of subtask size assigned to BS k by device n
x_n^0	percentage of subtask size of local computing by device n
$\mathbf{X}$	offloading strategy matrix
c_n^k	notation of connectivity between device n and BS k
$\mathbf{C}$	connectivity matrix between devices and BSs
$T_n^{0,\,\mathrm{co}}$	computing time of subtask of device n at local
β_n	density of task
l_n	size of device n's task
f_n	CPU frequency of device n
$T_n^{k,\,\mathrm{tr}}$	time of transmitting the subtask of device n to BS k
r_n^k	transmission rate of device n to the BS k
λ_n^k	proportion of computation resources allocated by BS k to device n
B_n^k	bandwidth of BS k
p_n	transmission power of device n
h_n^k	uplink channel gain of device n to BS k
σ	Gaussian noise power
$T_n^{k,\,\mathrm{co}}$	offloading computation time of device n to the BS k
f_k	CPU frequency of BS k
$T_n^{0,\,\mathrm{de}}$	local computation delay of device n
$T_n^{k,\,\mathrm{de}}$	offloading computation delay of device n
G_n^k	time of offloading the complete task of device n to BS k
T_n	processing time of device n's task
T_n^*	optimal processing time of device n's task
$\mathbf{X}_{-n}$	offloading strategy matrix of all devices except device n
$\mathcal{S}_n$	set of BSs that device n can offload
$\mathbf{x}_n^*$	optimal strategy of device n
$T_n^{k,\,\mathrm{de}\,*}$	minimum offloading delay of all subtasks of device n
$\mathbf{X}^*$	offloading strategy of all devices at Nash equilibrium

order to facilitate the modeling and analysis. It circumvents the complex transmission scheduling or radio resource allocation problem, which are not the main thrust of this article. In this way, the resulted transmission time can be regarded as an upper bound of the actual transmission time. Based on the Shannon Theorem, the transmission rate r_n^k from device n to BS k is expressed as

$$r_n^k = B_n^k \log_2\left(1 + \frac{p_n h_n^k}{\sigma}\right) \tag{2}$$

where B_n^k is the bandwidth of device n to the BS k, p_n is transmission power, h_n^k denotes the uplink channel gain, and the white Gaussian noise power is expressed by σ. Then, the transmission time of the subtask from device n to BS k is

$$T_n^{k,\mathrm{tr}} = \sum_{n\in\mathcal{N}} x_n^k \frac{l_n}{r_n^k}. \tag{3}$$

The proportion of computation resources allocated by BS k to device n is

$$\lambda_n^k = \frac{x_n^k l_n}{\sum_{n\in\mathcal{N}} x_n^k l_n}. \tag{4}$$

The subtask execution delay of device n at BS k is

$$\begin{aligned} T_n^{k,\mathrm{co}} &= x_n^k \frac{l_n \beta_n}{\lambda_n^k f_k} \\ &= \sum_{n\in\mathcal{N}} x_n^k \beta_n \frac{l_n}{f_k} \end{aligned} \tag{5}$$

where f_k denotes the CPU frequency of BS k.

Through the above analysis, we obtain the local offloading delay $T_n^{0,\mathrm{de}} = T_n^{0,\mathrm{co}}$, and the offloading computation delay is the sum of the transmission time and execution time. The offloading computation delay is expressed as

$$\begin{aligned} T_n^{k,\mathrm{de}} &= \sum_{n\in\mathcal{N}} x_n^k \frac{l_n}{r_n^k} + \sum_{n\in\mathcal{N}} x_n^k \beta_n \frac{l_n}{f_k} \\ &= \sum_{n\in\mathcal{N}} x_n^k l_n \left(\frac{1}{r_n^k} + \frac{\beta_n}{f_k}\right). \end{aligned} \tag{6}$$

Since all subtasks of device n are processed in parallel, we choose the longest execution time among all subtasks as the final execution time of device n, expressed as

$$T_n(\mathbf{x}_n, \mathbf{X}_{-n}) = \max_{k\in\{0\}\cup\mathcal{K}} \left\{T_n^{k,de}\right\} \tag{7}$$

where $\mathbf{X}_{-n} = (\mathbf{x}_1^\mathsf{T}, \ldots, \mathbf{x}_{n-1}^\mathsf{T}, \mathbf{x}_{n+1}^\mathsf{T}, \ldots, \mathbf{x}_N^\mathsf{T})^\mathsf{T}$ is the offloading strategy matrix except device n.

C. Problem Formulation

To minimize the offloading delay of each individual device, the following optimization problem can be formulated as:

$$\min_{\mathbf{x}_n} \quad T_n \tag{8}$$

$$\text{s.t.} \quad T_n^{0,\mathrm{de}} \le T_n \tag{8a}$$

$$T_n^{k,\mathrm{de}} \le T_n \tag{8b}$$

$$\sum_{n\in\mathcal{N}} \lambda_n^k \le 1 \quad \forall k \in \mathcal{K} \tag{8c}$$

$$x_n^0 + \sum_{k\in\mathcal{K}} x_n^k = 1 \tag{8d}$$

$$x_n^0, x_n^k \ge 0 \quad \forall k \in \mathcal{K} \tag{8e}$$

$$x_n^k \le c_n^k \quad \forall k \in \mathcal{K}. \tag{8f}$$

Constraints (8a) and (8b) indicate that the local computation delay and the offloading computation delay of device n do not exceed the task completion time. Constraint (8c) guarantees that the computation resources allocated to the devices are no more than the total resources of the BSs. Constraints (8d) and (8e) ensure that the subtasks are collaboratively executed by the devices and the BSs. Constraint (8f) indicates that the subtasks of the device can only be offloaded to its connected BSs. $c_n^k = 0$ represents that there is no connection between device n and BS k, and thus, $x_n^k = 0$. The condition $c_n^k = 1$ means that the device n is in the coverage area of the BS k. Constraint (8f) ensures that the devices can only offload subtasks to the connected BSs.

According to the above communication and computation models, it is obvious that the computation offloading strategies $\mathbf{X} = (\mathbf{x}_1^\mathsf{T}, \mathbf{x}_2^\mathsf{T}, \ldots, \mathbf{x}_N^\mathsf{T})^\mathsf{T}$ among the devices are coupled. The optimization problem is nonlinear and nonconvex. Thus, it is challenging to solve the offloading problem efficiently. We will propose a game-theoretic approach to solve the problem in the following section.

IV. Distributed Task Offloading

In this section, we exploit a game-theoretic approach to efficiently solve the offloading problem for all the devices. Specifically, in Section IV-A, we analyze the property of the offloading problem, and then, propose the offloading strategy for each device. In Section IV-B, we build the game model, prove the existence of generalized Nash equilibrium (GNE), and propose the DOLA for all the devices to reach the Nash equilibrium.

A. Offloading Strategy for Each IoT Device

1) Property Analysis of the Offloading Problem: The subtasks of each device n are executed in parallel. Thus, we aim to keep the largest delay of the subtasks as small as possible, and the subtasks will be transferred from the BSs with heavy workload to other BSs with relatively low workload. Finally, the delays of all the subtasks will be same, as proved in Theorem 1.

Theorem 1: The local computation delay is equal to the offloading computation delay of the subtasks when the device's offloading policy is optimal, which can be expressed as

$$T_n^{k,\mathrm{de}} = T_n = \min_{\mathbf{x}_n}\left\{\max_{k\in\{0\}\cup\mathcal{K}}\left\{T_n^{k,de}\right\}\right\} \quad \forall k \in \{0\}\cup\mathcal{K}, x_n^k > 0. \tag{9}$$

Proof: We consider that $\mathbf{x}_n^*$ is the optimal offloading policy for the device n. If (9) does not hold, then, $\exists k \in \mathcal{K}, T_n^{k,\mathrm{de}^*} = T_n$, and $T_n^{0,\mathrm{de}^*} < T_n^{k,\mathrm{de}^*}$. Furthermore, the local computation delay is

$$T_n^{0,\mathrm{de}^*} = x_n^{0^*}\beta_n\frac{l_n}{f_n} \tag{10}$$

and the offloading computation delay is

$$\begin{aligned} T_n^{k,\mathrm{de}^*} &= \sum_{n\in\mathcal{N}} x_n^{k^*}\frac{l_n}{r_n^k} + \sum_{n\in\mathcal{N}} x_n^{k^*}\beta_n\frac{l_n}{f_k} \\ &= \sum_{n\neq m} x_m^k\frac{l_m}{r_m^k} + x_n^{k^*}\frac{l_n}{r_n^k} + \sum_{n\neq m} x_m^k\beta_m\frac{l_m}{f_k} \\ &\quad + x_m^{k^*}\beta_n\frac{l_n}{f_k}. \end{aligned} \tag{11}$$

There exists a task transition factor $\Delta x_0^{k^*} > 0$, which makes the following three expressions [i.e., (12), (13), and (14)] hold:

$$\begin{aligned} \overline{T_n^{k,\mathrm{de}^*}} &= \sum_{n\neq m} x_m^k\frac{l_m}{r_m^k} + \left(x_n^{k^*} - \Delta x_0^{k^*}\right)\frac{l_n}{r_n^k} \\ &\quad + \sum_{n\neq m} x_m^k\beta_m\frac{l_m}{f_k} + \left(x_m^{k^*} - \Delta x_0^{k^*}\right)\beta_n\frac{l_n}{f_k} \\ &< T_n^{k,\mathrm{de}^*} \end{aligned} \tag{12}$$

$$\overline{T_n^{k,\mathrm{de}^*}} = \left(x_n^{0^*} + \Delta x_0^{k^*}\right)\beta_n\frac{l_n}{f_n} < T_n^{0,\mathrm{de}^*} \tag{13}$$

$$\overline{T_n^{k,\mathrm{de}^*}} > \overline{T_n^{0,\mathrm{de}^*}}. \tag{14}$$

After transferring subtasks, the offloading strategy of device n also satisfies constraints (8d) and (8e) as $x_m^{k^*} - \Delta x_0^{k^*} > 0$, $x_n^{0^*} + \Delta x_0^{k^*} > 0$ and $(x_n^{0^*} + \Delta x_0^{k^*}) + x_n^1 + \cdots + (x_m^{k^*} - \Delta x_0^{k^*}) + \cdots + x_n^K = 1$.

When $T_n^{0,\mathrm{de}^*} = T_n$, there exists a task transition factor $\Delta x_0^{k^*} > 0$, which can further reduce T_n^{k,de^*}, and the result can be proved in the similar way. ■

2) Offloading Size Decision: For each device n, given the set of BSs selected for task offloading, we can obtain the offloading strategy and the execution time of device n according to Theorem 1. We use $\mathcal{S}_n = \{k \in \mathcal{K} \mid x_n^k > 0\}$ to denote the set of BSs that device n can offload to. The offloading decision of device n is $\mathbf{x}_n$, and we define $G_n^k = l_n([1/r_n^k] + [\beta_n/f_k])$. From (1), (6), and (9), we obtain

$$\sum_{m\neq n} x_m^k G_m^k + x_n^k G_n^k = x_n^0\frac{l_n\beta_n}{f_n} \tag{15}$$

$$x_n^0 + \sum_{k\in\mathcal{S}_n} x_n^k = 1. \tag{16}$$

Hence

$$\sum_{m\neq n} x_m^k G_m^k + x_n^k G_n^k = \left(1 - \sum_{k\in\mathcal{S}_n} x_n^k\right)\beta_n\frac{l_n}{f_n}. \tag{17}$$

Then

$$\begin{aligned} &\left(G_n^k + \frac{l_n\beta_n}{f_n}\right)x_n^k + \frac{l_n\beta_n}{f_n}\sum_{i\in\mathcal{S}_n\setminus k} x_n^i \\ &= \frac{l_n\beta_n}{f_n} - \sum_{m\neq n} x_m^k G_m^k. \end{aligned} \tag{18}$$

Based on the BSs, which the subtasks of device n can be offloaded to, we obtain a set of linear equations expressed as (19), shown at the bottom of the next page, and it has a total of $|\mathcal{S}_n|$ linear equations. We then calculate the fraction of task of device n offloaded to BS k as (20), shown at the bottom of the next page, and the fraction of task computed by device n locally is

$$x_n^0(\mathbf{X}_{-n}) = \frac{1 + \sum_{l\in\mathcal{S}_n}\frac{\sum_{m\neq n} x_m^l G_m^l}{G_n^l}}{1 + \frac{l_n\beta_n}{f_n}\sum_{l\in\mathcal{S}_n}}. \tag{21}$$

Furthermore, from (9), the offloading computation delay $T_n(\mathbf{X}_{-n})$ is equal to the local computation. Then

$$T_n(\mathbf{X}_{-n}) = T_n^{0,\mathrm{de}}. \tag{22}$$

Because the sum of the percentages of all subtasks is 1, taking advantage of (8c), we obtain

$$T_n(\mathbf{X}_{-n}) = 1 - \sum_{k\in\mathcal{S}_n} x_n^k(\mathbf{X}_{-n})\frac{l_n\beta_n}{f_n}. \tag{23}$$

Finally, the execution time is

$$T_n(\mathbf{X}_{-n}) = \frac{1+\sum_{l\in\mathcal{S}_n}\frac{T_{-n}^{k,\mathrm{de}}}{G_n^l}}{1+\frac{l_n\beta_n}{f_n}\sum_{l\in\mathcal{S}_n}\frac{1}{G_n^l}}\frac{l_n\beta_n}{f_n} \tag{24}$$

where $T_{-n}^{k,\mathrm{de}} = \sum_{m\neq n} x_m^k G_m^k$ is the sum of the subtask offloading delays of all devices on BS k except the subtask of device n.

3) BS Selection: Device n cannot offload its subtasks to all the connected BSs. This is because some connected BSs may be already heavy loaded, and offloading subtasks to these BSs will degrade the performance. We propose Theorem 2, which is useful for obtaining $|\mathcal{S}_n|$.

Theorem 2: For device n, if the delay sum of all the subtasks (excluding the subtask from device n) on BS k is smaller than the delay of local computing, then device n can select BS k for offloading.

Proof: Based on Theorem 1, we can obtain the offloading delay in the last iteration for device n. Since the local computation delay is equal to the offloading delay under the optimal solution, the subtasks of device n can only be offloaded to the BSs whose task execution time is less than the local computation delay. ■

Based on Theorem 2, $|\mathcal{S}_n|$ can be calculated by

$$\mathcal{S}_n = \left\{k \in \mathcal{K} \mid T_{-n}^{k,\mathrm{de}} < T_n(\mathbf{X}_{-n})\right\} \tag{25}$$

where $T_{-n}^{1,\mathrm{de}} \le T_{-n}^{2,\mathrm{de}} \le \cdots \le T_{-n}^{K,\mathrm{de}}$, which means that the task execution time of all BSs is arranged in a nonincreasing order, and $T_{-n}^{k,\mathrm{de}} = \infty$ if BS k is not connected to device n. The device n needs to go through all candidate BSs (subset of the connected BSs of device n with $c_n^k = 1$), and then, select a set of BSs. During this period, device n sorts $T_{-n}^{k,\mathrm{de}}(t)$ first and then selects the appropriate BSs. Let $M = \sum_k c_n^k$, and the computational complexity in the worst case is $O(M\log M)$. With some advanced sorting algorithms (e.g., Radix sort [31]), the complexity can be further reduced to $O(M)$. In reality, the value of M is relatively small. Therefore, the extra delay for making decision in our approach can be negligible.

B. Distributed Task Offloading Algorithm Design

We then propose the task offloading algorithm for the devices. As the devices compete for the limited resources and each cares for its own interests, we formulate an offloading game model for the devices, and prove the existence of Nash equilibrium. Then, we propose the offloading algorithm.

1) Game Model: For each device n, given other devices' offloading policy $\mathbf{X}_{-n}$, it needs to make its own offloading decision $\mathbf{x}_n = (x_n^0, x_n^1, \ldots, x_n^K)$, $x_n^k \in [0,1]$. The goal is to minimize the execution time expressed as

$$\min_{\mathbf{x}_n}\ T_n(\mathbf{x}_n, \mathbf{X}_{-n}) \quad \forall n \in \mathcal{N}. \tag{26}$$

We formulate the above problem as a strategic game $\Gamma = (\mathcal{N}, \{\mathbf{a}_n\}_{n\in\mathcal{N}}, \{T_n\}_{n\in\mathcal{N}})$, where $\mathcal{N}$ is the set of players, $\mathbf{a}_n = [0,1]^{K+1}$ is the set of strategies for player n, and $\mathbf{x}_n \in \mathbf{a}_n$, $\{T_n\}_{n\in\mathcal{N}}$ is the player n's cost function to be minimized. The desired offloading strategy is the Nash equilibrium solution defined in Definition 1.

Definition 1: $\mathbf{X}^*$ is denoted as a Nash equilibrium of the offloading game, if there is no way for any player to further reduce the cost by one-side changing its decision under the state of Nash equilibrium $\mathbf{X}^*$, i.e.,

$$T_n\big(\mathbf{x}_n^*, \mathbf{X}_{-n}^*\big) < T_n\big(\mathbf{x}_n, \mathbf{X}_{-n}^*\big) \quad \forall n \in \mathcal{N}. \tag{27}$$

Each device makes the corresponding decision for its own benefit. Under the mutual influence of other devices, each device n in the state of GNE can get a satisfactory offloading strategy.

2) Existence of Generalized Nash Equilibrium: We now prove the existence of GNE. Define $x_n^* \in \Psi_n(\mathbf{X}_{-n}^*)$, where $\Psi_n(\mathbf{X}_{-n}^*)$ is the optimal strategy of device n, and $\Psi(\mathbf{X})$ is the holistic optimization strategy of all devices expressed as

$$\Psi(\mathbf{X}) = \Psi_1\big(\mathbf{X}_{-1}^*\big) \otimes \Psi_2\big(\mathbf{X}_{-2}^*\big) \otimes \cdots \otimes \Psi_N\big(\mathbf{X}_{-N}^*\big) \tag{28}$$

where $\mathbf{X} = \mathbf{x}_1^* \otimes \mathbf{x}_2^* \otimes \cdots \otimes \mathbf{x}_N^*$, and $\otimes$ is the Cartesian product. Then, we prove that there exists a GNE point of the optimization problem.

Lemma 1: The optimal strategy set $\mathbf{X}$ is convex.

Proof: We set $\mathbf{x}_1, \mathbf{x}_2 \in \mathbf{a}_n$, and $\mu \in [0,1]$. According to the definition of convex set, $\mu\mathbf{x}_1 + (1-\mu)\mathbf{x}_2 \in \mathbf{a}_n$ always

$$\begin{bmatrix} G_n^1 + \frac{l_n\beta_n}{f_n} & \frac{l_n\beta_n}{f_n} & \cdots & \frac{l_n\beta_n}{f_n} \\ \frac{l_n\beta_n}{f_n} & G_n^2 + \frac{l_n\beta_n}{f_n} & \cdots & \frac{l_n\beta_n}{f_n} \\ \vdots & & \ddots & \\ \frac{l_n\beta_n}{f_n} & \cdots & & G_n^M + \frac{l_n\beta_n}{f_n} \end{bmatrix} \begin{bmatrix} x_n^1 \\ x_n^2 \\ \vdots \\ x_n^M \end{bmatrix} = \begin{bmatrix} \frac{l_n\beta_n}{f_n} - \sum_{m\neq n} x_m^1 G_m^1 \\ \frac{l_n\beta_n}{f_n} - \sum_{m\neq n} x_m^2 G_m^2 \\ \vdots \\ \frac{l_n\beta_n}{f_n} - \sum_{m\neq n} x_m^M G_m^M, \end{bmatrix} \tag{19}$$

$$x_n^k(\mathbf{X}_{-n}) = \begin{cases} \frac{1}{G_n^k}\left(\frac{l_n\beta_n}{f_n} - T_{-n}^{k,\mathrm{de}}\right) - \frac{\frac{1}{G_n^k}\frac{l_n\beta_n}{f_n}\sum_{l\in\mathcal{S}_n}\frac{1}{G_n^l}\left(\frac{l_n\beta_n}{f_n} - \sum_{m\neq n} x_m^l G_m^l\right)}{1+\frac{l_n\beta_n}{f_n}\sum_{l\in\mathcal{S}_n}\frac{1}{G_n^l}}, & T_{-n}^{k,\mathrm{de}} < T_n(\mathbf{X}_{-n}) \\ 0, & T_{-n}^{k,\mathrm{de}} \ge T_n(\mathbf{X}_{-n}) \end{cases} \tag{20}$$

Algorithm 1: DOLA

1 Initialize the iterative epoch t=0.
2 **for** $n \in \mathcal{N}$ **do**
3 Initialize $x_n^0(t) = 1$, i.e., each device n selects local computing.
4 **repeat**
5 **for** *all devices in parallel* **do**
6 Send TRI to BSs, including the size of task l_n, the computation frequency f_n, and the transmit power p_n.
7 Receive the values of G_n^k and $T_{-n}^{k,\text{de}}(t)$ from the connected BSs.
8 Select the set of BSs $\mathcal{S}_n(t)$ based on Theorem 2 and (25).
9 Compute the optimal offloading strategy $\mathbf{x}_n^*(t)$ and execution time $T_n^*(t)$ based on (20), (21) and (24).
10 **if** $\Delta = T_n(t) - T_n^*(t) > \varepsilon$ **then**
11 Send the UR message to BSs requesting for the strategy update opportunity.
12 **if** *BSs accept UR request* **then**
13 Update the strategy $\mathbf{x}_n(t+1) = \mathbf{x}_n(t)^*$ for the next epoch.
14 **else**
15 Remain $\mathbf{x}_n(t+1) = \mathbf{x}_n(t)$ for the next epoch.
16 **else**
17 Remain $\mathbf{x}_n(t+1) = \mathbf{x}_n(t)$ for the next epoch.
18 $t = t + 1$.
19 **until** *No RTU messages are broadcasted for M continuous epochs*;

holds. Therefore, $\mathbf{a}_n$ is a convex set. $\mathbf{X}$ is the Cartesian product of $\mathbf{x}_n^* \in \mathbf{a}_n$. Thus, the set $\mathbf{X}$ is convex. ■

Lemma 2: The optimal strategy set $\mathbf{X}$ is compact.

Proof: We set $\mathbf{x} \in \mathbf{a}_n$. Since $\mathbf{a}_n = [0, 1]^{K+1}$ and $\|\mathbf{x}\|_2 \leq N$, $\mathbf{x}$ is a bounded and closed set. $\mathbf{X}$ is the Cartesian product of $\mathbf{x}_n^*$, $\mathbf{x}_n^* \in \mathbf{a}_n$. Therefore, the set $\mathbf{X}$ is compact. ■

Based on Lemmas 1 and 2, we further propose the existence of GNE as expressed in Theorem 3.

Theorem 3: The offloading game always has a Nash equilibrium and the finite improvement property.

Proof: The set $\mathbf{X}$ is convex and compact according to Lemmas 1 and 2, and it is also continuous based on (20) and (21). Thus, $\Psi(\mathbf{X})$ must have a fixed point on $\mathbf{X}$ based on Brouwer's fixed-point theorem [26], [32], and the point is a GNE of the optimization problem. ■

3) Distributed Offloading Algorithm: We propose the DOLA as shown in Algorithm 1. t stands for the integrative epoch ID. For each device n, we initialize the offloading decision as $x_n^0(0) = 1$, i.e., each device n chooses local offloading. Then, each device n sends task-related information (TRI) to the connected BSs, including transmission power p_n, size of task l_n and density β_n, in line 6. The BSs calculate the expression $G_n^k = l_n([1/r_n^k] + [\beta_n/f_k])$ and the current task computation time $T_{-n}^{k,\text{de}}(t) = \sum_{m \neq n} x_m^k(t) G_m^k$ for device n, and return the values to device n in line 7. In line 9, device n obtains the set $\mathcal{S}_n(t)$ of selected BSs according to Theorem 2 and (25). In line 9, device n obtains the optimal offloading strategy $\mathbf{x}_n^*(t)$ and execution time $T_n^*(t)$ based on (20), (21), and (24).

Then, device n updates its offloading strategy if it can further reduce the execution time by ε, as stated in line 11. Device n broadcasts the update request (UR) message to the BSs and requests for updating the offloading decision. If the request is accepted by the BSs, the BSs broadcast the update permission (UP) message to device n, and then, the device n updates its strategy. Otherwise, device n keeps the current strategy for next epoch. If device n cannot reduce its execution time by ε, the device keeps the same strategy for the next epoch.

Finally, if there is no RTU message sent to the BSs for M continuous epochs, it means no single device can further reduce the execution time. Then, the BSs broadcast the END message to all devices.

TABLE II
EXPERIMENT SETUP

Parameters	Value
K (number of BSs)	125
B (bandwidth of BS)	10 MHz
σ (background noise)	-100 dBm
N (number of devices)	825
l_n (size of task from device n)	$U \sim [5, 10]$ MB
f_n (computation capability of device)	1 GHz
f_k (computation capability of BS)	4 GHz
β (density of task)	50 cycles/bit
ε (threshold variable)	0.001

V. EXPERIMENTAL EVALUATION

A. Setup

We adopt the real-world data set from Melbourne CBD area of Australia to validate our DOLA algorithm. The data set includes the locations of 125 BSs and 825 devices in the area [33]. Fig. 2 shows the distributions of these 125 BSs and 825 devices. The blue points represent the locations of devices and the red points represent the locations of the BSs. The red circles represent the coverage areas of the BSs. It can be seen from Fig. 2 that the deployment of BSs is dense, and there are a huge number of overlapping areas among these BSs.

For each BS, the coverage radius is set as 200 m, the channel bandwidth is 10 MHz, the background noise is -100 dBm, and the computation capability of each BS is 4 GHz [24]. For each task generated by the device, the task size is randomly set between 5 and 10 MB, and the density is 50 cycles/bit. The computation capability of each device is 1 GHz and the transmission power is 200 mW. The channel gain is set to d_n^{k-3}, where d_n^k is the distance between device n and BS k. The threshold parameter ε is set to 0.001. The detailed parameter settings are listed in Table II.

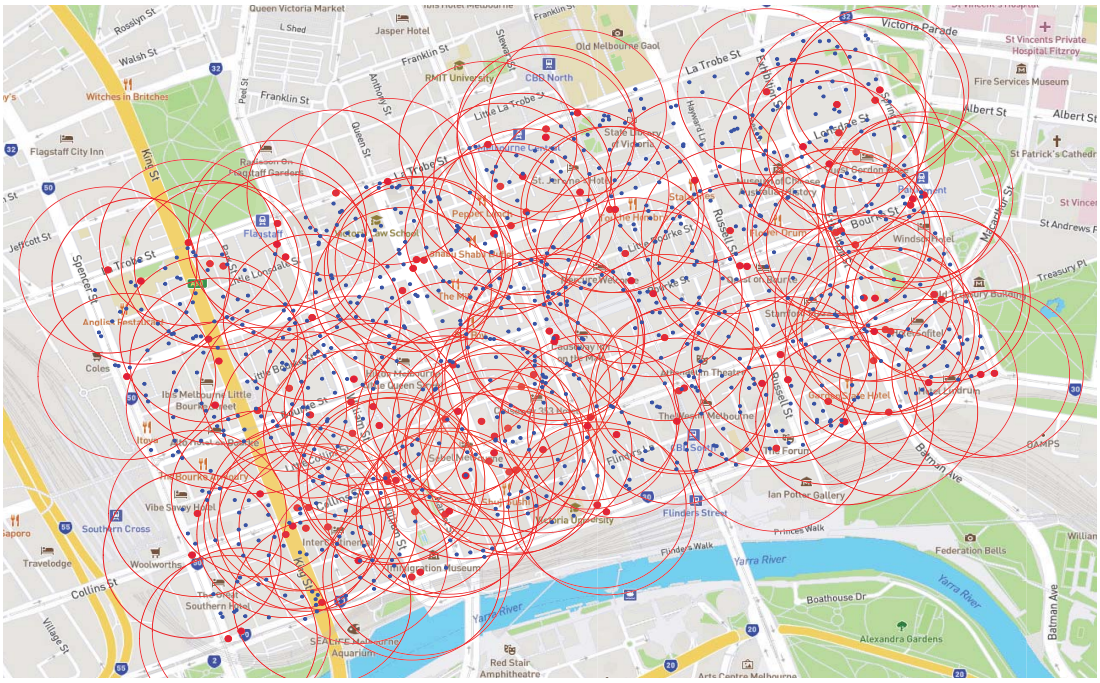

Fig. 2. Distribution of BSs and devices in Melbourne CBD area of Australia.

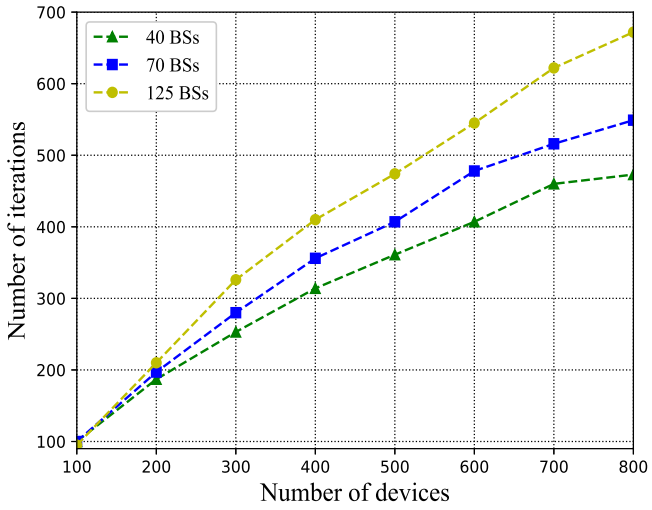

Fig. 3. Number of iterations with different numbers of devices.

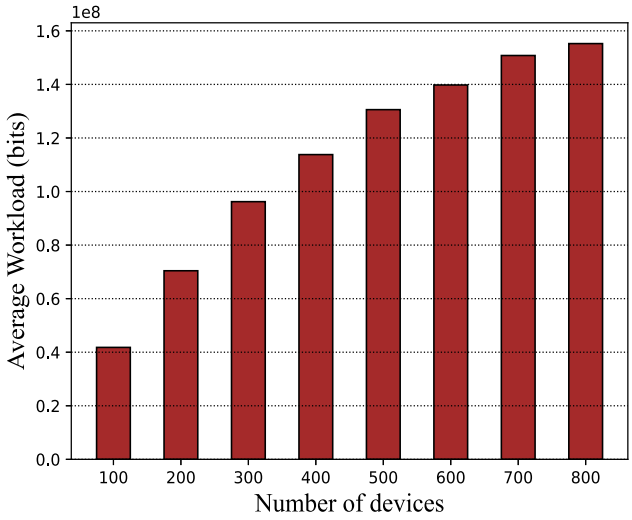

Fig. 4. Average workload with different devices.

B. Parameter Analysis

Fig. 3 shows the required number of iterations to reach the Nash equilibrium with different numbers of devices and different numbers of BSs. The number of devices is varied from 100 to 800, and there are three settings of the BS number, i.e., 40, 70, and 125. We can see that for all the three BS number settings, as the number of devices increases, the number of iterations to reach the Nash equilibrium also grows. This is because the solution space size grows rapidly and the overall offloading problem becomes more complex with the rise of device number, increasing the number of iterations. Nevertheless, compared with the exponential growth speed of the solution space size with the number of devices, we can find from Fig. 3 that the growth speed of the number of iterations is slower than linear growth. This shows the scalability of our proposed DOLA algorithm, which is important for real-world and large-scale system.

Fig. 4 shows the average workload (in bits) of each BS with different numbers of devices. Similarly, the number of devices is varied from 100 to 800. Obviously, the average workload of each BS increases with the increase of device number, as this brings more workload to the whole system. More specifically, it can be observed that when the number of devices is less than 300, the average workload of each BS rises linearly with the increase of device number. This is because when the device number is small, the overall workload in the system is relatively low, and most of the BSs are not heavily loaded or with high utilization. In this case, as the workload from the devices

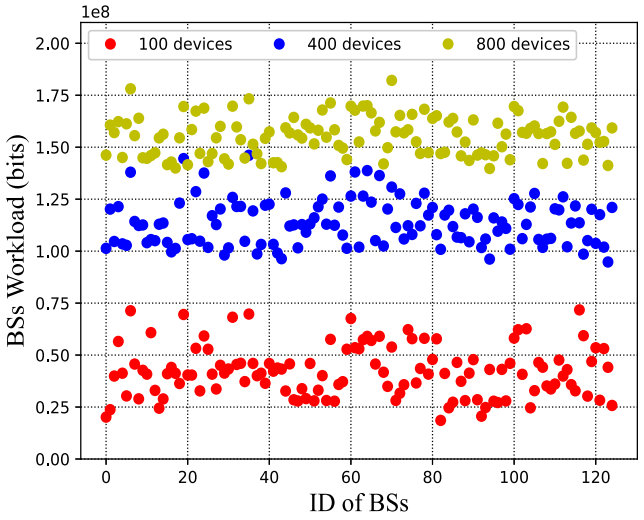

Fig. 5. Workload on different BSs.

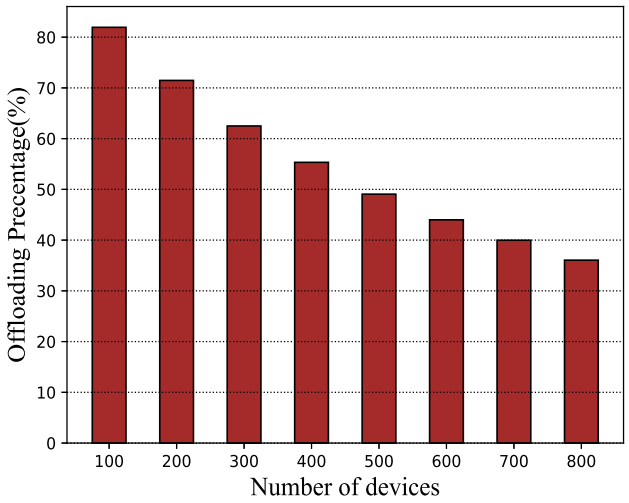

Fig. 6. Offloading percentage on different devices.

grows, the BSs have sufficient computation resources to handle the workload. Thus, the BSs would admit the tasks offloaded from the devices as many as possible. Nevertheless, it is seen that when the device number is increased further (from about 400 in Fig. 4), the growth of each BS's workload would slow down, and the growth speed is slower than linear growth (the device number's growth speed). Actually, as the device number increases further, more and more tasks are generated, and the BSs in the system would face resource limitations. Therefore, the BSs would reduce the admitted workload to maintain the performance. This also demonstrates the adaptive property of our proposed DOLA algorithm.

To further evaluate the offloaded workload of each BS with more details, Fig. 5 illustrates the workload of each of the 125 BSs with different device numbers. Three different settings of device numbers are considered, i.e., 100, 400, and 800. We can find that when the device number rises, the workload of each BS also rises. Especially, when the device number is 100, the workload of each BS is between 0.2×10^8 bits and 0.7×10^8 bits. Besides, the workload distribution is relatively sparse as some BSs are not heavy loaded or in high utilization, and these BSs only have a relatively small number of tasks to process. When the number of device is 800, the workload of each BS is between 1.4×10^8 and 1.8×10^8 bits, and the workload distribution is dense. This is because when the device number is further increased, more and more tasks are generated, making the BSs heavy loaded. It is also observed that for each of the three settings of the device number, the workload of different BSs generally stays in the same horizontal region. This shows that our proposed DOLA algorithm can balance the workload among different BSs. Actually, load balance is an important factor for maintaining system stability in workload scheduling, which avoids some BSs from very high utilization and even crash.

Finally, Fig. 6 illustrates the average offloading percentages of the devices with varied device number. It is shown that as the number of devices rises, the average offloading percentage decreases and the decreasing rate becomes slower when the number of the devices is increased further (from about 400 in Fig. 6). Especially, when there are 500 devices, the average offloading rate of devices is 49%. When there are 800 devices, the average offloading rate falls to 37%.

C. Comparison Experiments

In this part, we carry out comparison experiments to further evaluate the performance of our DOLA algorithm. The following three benchmark experiments are adopted.

1) *Local Computing:* Each device performs the task locally. No tasks are offloaded to the BSs for processing.
2) *Total Offloading Computing:* All the tasks are offloaded to the BSs for processing. No tasks are processed locally on the devices.
3) *MCF:* To further validate the efficiency and effectiveness of our proposed approach, we also select a state-of-the-art approach for comparison, namely, MCF from [34].
4) *Equal-Sized Subtasks Offloading:* For each device n, we divide the task size equally by $\sum_k c_n^k + 1$. Here, $\sum_k c_n^k$ stands for the number of connected BSs for device n, and 1 stands for local computing. The $(\sum_k c_n^k + 1)$ subtasks are processed on the connected BSs and locally in parallel.

Fig. 7 shows the average delays (in seconds) of the five algorithms with different device numbers. We can see that when the device number rises, the delay of the local computing algorithm stays almost the same, while the delays of the other four algorithm continue to increase. Nevertheless, the increasing rate of our DOLA's delay is smaller than linear increasing rate (the increasing rate of the device number), while the delays of the local computing and equal-sized subtasks offloading algorithms show a linear increasing rate.

Specifically, when the device number is smaller than 500, the delay of the local computing algorithm is the smallest among the five algorithms, and the delays of our DOLA algorithm and the equal-sized subtasks offloading algorithm are similar. This is because when the device number is small, the workload generated by these devices is relatively low. In this case, the BSs have sufficient computation resources to handle the workload from the devices. When the device number is

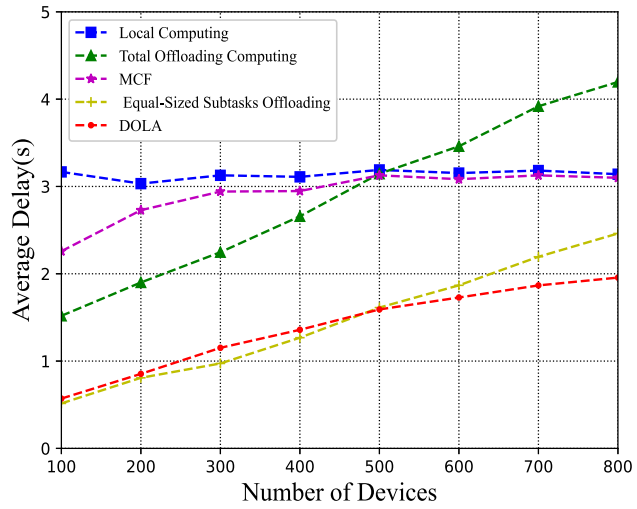

Fig. 7. System average delay with different numbers of BSs.

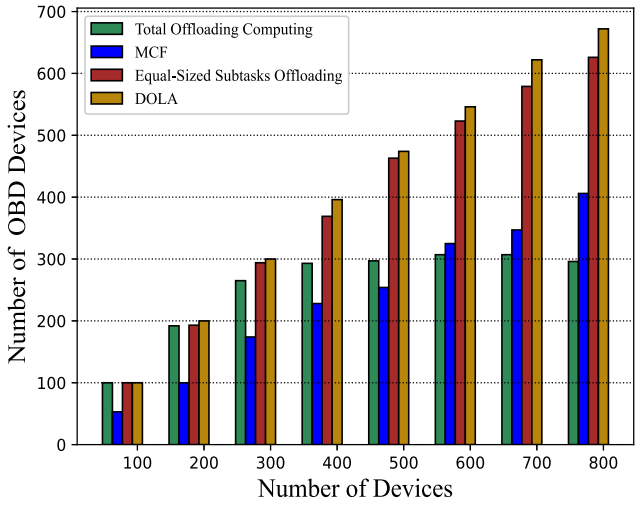

Fig. 8. Number of OBD devices with different numbers of devices.

further increased (more than 500 in Fig. 7), the superiority of our DOLA algorithm is more obvious. In this case, DOLA's delay is the smallest among the five algorithms. The delay of the total offloading computing algorithm is the longest. This is because when the device number is too large, these devices generate a large number of tasks, and the computing resources of the BSs are not sufficient to process all the tasks. Thus, offloading all the tasks to the BSs actually enhances the delay and degrades the performance. The results in Fig. 7 demonstrate that our algorithm can effectively reduce the execution time.

To evaluate the benefits of offloading toward local computing, we define the concept of offloading beneficial device (OBD). A device is called an OBD device if the delay of integrating offloading control decisions is smaller than the delay of solely local computing, and the device benefits from offloading. Fig. 8 shows the number of OBD devices of the total offloading computing, the equal-sized subtasks offloading, and the DOLA algorithms with different device number. The local computing algorithm is skipped for comparison as its OBD device number is 0. We can see that the OBD device number of our DOLA algorithm is always the largest, and the OBD device number of the total offloading computing algorithm is always the smallest. The results in Fig. 8 demonstrate the increase of the number of beneficial devices compared to the MCF algorithm.

To be more specific, when the device number is less than 300, the OBD device numbers of all the four algorithms rise with the increase of device number. This is because the computing resources and capacities of the BSs are relatively sufficient for the devices, and the devices do not need to compete for the sufficient resources. When the device number is increased further (from 300 in Fig. 8), the OBD device numbers of our DOLA algorithm and the equal-sized subtasks offloading algorithm both rise. What is more, we can find that the advantage of our proposed DOLA algorithm toward the equal-sized subtasks offloading algorithm is more obvious, and the gap of the OBD device numbers between our DOLA and the equal-sized subtasks offloading algorithms is larger. We also observe that the OBD device number of the total offloading computing algorithm cannot improve anymore and stays the same. This is because when the device number is increased further, the computing resources and capacities of the BSs are not sufficient enough to deal with all the generated tasks of the devices. The total offloading computing algorithm accepts all the tasks generated on the devices, resulting in large delay and poor performance.

VI. Conclusion

In this article, we have investigated a MEC system with divisible tasks generated from IoT devices in the overlapping coverage area of multiple BSs. The divisible tasks have been divided into several subtasks, and executed locally or offloaded to multiple BSs for processing in parallel. We have formulated the delay optimal offloading problem, and theoretically, analyzed the properties of the offloading problem. We have also proved the existence of Nash equilibrium and proposed the DOLA algorithm for all the IoT devices to reach the equilibrium. The DOLA algorithm can obtain the satisfying equilibrium offloading strategies for the devices in parallel. Finally, we have adopted the real-world data from Melbourne CBD area of Australia for evaluation. The experimental results have demonstrated that our DOLA algorithm can converge quickly and is adaptive when the system scale increases. For our future work, we will consider the impact of energy consumption of devices as most IoT devices are battery limited. Since the energy consumption and delay are two conflicting metrics, we will study the joint optimization of energy and delay to achieve the tradeoff between those two metrics.

References

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

[2] N. Y. Philip, J. J. P. C. Rodrigues, H. Wang, S. J. Fong, and J. Chen, "Internet of Things for in-home health monitoring systems: Current advances, challenges and future directions," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 300–310, Feb. 2021.

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

[4] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[5] Z. Chang, L. Liu, X. Guo, and Q. Sheng, "Dynamic resource allocation and computation offloading for IoT fog computing system," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3348–3357, May 2021.

[6] F. Tang, B. Mao, N. Kato, and G. Gui, "Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 2027–2057, 3rd Quart., 2021.

[7] J. G. Andrews *et al.*, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[8] Y. Xu, G. Gui, H. Gacanin, and F. Adachi, "A survey on resource allocation for 5G heterogeneous networks: Current research, future trends, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 668–695, 2nd Quart., 2021.

[9] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing–A key technology towards 5G," Eur. Telecommun. Stand. Inst., Sophia Antipolis, France, Rep. no. 11, 2015.

[10] D. Zeng, L. Gu, S. Pan, J. Cai, and S. Guo, "Resource management at the network edge: A deep reinforcement learning approach," *IEEE Netw.*, vol. 33, no. 3, pp. 26–33, May/Jun. 2019.

[11] M. Dai, Z. Su, J. Li, and J. Zhou, "An energy-efficient edge offloading scheme for UAV-assisted Internet of Things," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2020, pp. 1293–1297.

[12] 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.

[13] 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 Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[14] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019.

[15] Y. Wu, L. P. Qian, K. Ni, C. Zhang, and X. Shen, "Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 3, pp. 392–407, Jun. 2019.

[16] M.-H. Chen, B. Liang, and M. Dong, "A semidefinite relaxation approach to mobile cloud offloading with computing access point," in *Proc. IEEE 16th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2015, pp. 186–190.

[17] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

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

[19] L. Zhang, R. Chai, T. Yang, and Q. Chen, "Min-max worst-case design for computation offloading in multi-user MEC system," in *Proc. IEEE INFOCOM IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2020, pp. 1075–1080.

[20] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber-wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, May 2018.

[21] J. Zhou, D. Tian, Z. Sheng, X. Duan, and X. Shen, "Distributed task offloading optimization with queueing dynamics in multiagent mobile-edge computing networks," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12311–12328, Aug. 2021.

[22] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4188–4200, Jun. 2019.

[23] H. Q. Le, H. Al-Shatri, and A. Klein, "Efficient resource allocation in mobile-edge computation offloading: Completion time minimization," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 2513–2517.

[24] T. Mahn, H. Al-Shatri, and A. Klein, "Distributed algorithm for energy efficient joint cloud and edge computing with splittable tasks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2019, pp. 1–7.

[25] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "DATS: Dispersive stable task scheduling in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3423–3436, Apr. 2019.

[26] Z. Liu, Y. Yang, K. Wang, Z. Shao, and J. Zhang, "POST: Parallel offloading of splittable tasks in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3170–3183, Apr. 2020.

[27] G. R. MacCartney and T. S. Rappaport, "Millimeter-wave base station diversity for 5G coordinated multipoint (CoMP) applications," *IEEE Trans. Wireless Commun.*, vol. 18, no. 7, pp. 3395–3410, Jul. 2019.

[28] M. K. Karakayali, G. J. Foschini, and R. A. Valenzuela, "Network coordination for spectrally efficient communications in cellular systems," *IEEE Wireless Commun.*, vol. 13, no. 4, pp. 56–61, Aug. 2006.

[29] D. Nowak, T. Mahn, H. Al-Shatri, A. Schwartz, and A. Klein, "A generalized nash game for mobile edge computation offloading," in *Proc. 6th IEEE Int. Conf. Mobile Cloud Comput. Services Eng. (MobileCloud)*, 2018, pp. 95–102.

[30] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[31] O. Polychroniou and K. A. Ross, "A comprehensive study of main-memory partitioning and its application to large-scale comparison-and radix-sort," in *Proc. ACM SIGMOD Int. Conf. Manag. data*, 2014, pp. 755–766.

[32] G. Debreu, "A social equilibrium existence theorem," *Proc. Nat. Acad. Sci.*, vol. 38, no. 10, pp. 886–893, 1952.

[33] Q. Peng *et al.*, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2019, pp. 91–98.

[34] P. Lai *et al.*, "Cost-effective app user allocation in an edge computing environment," *IEEE Trans. Cloud Comput.*, early access, Jun. 11, 2020, doi: 10.1109/TCC.2020.3001570.