Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9977700, 13 pages
https://doi.org/10.1155/2021/9977700

Research Article

Cooperative Offloading in D2D-Enabled Three-Tier MEC Networks for IoT

Jingyan Wu,[1,2] **Jiawei Zhang**,[1] **Yuming Xiao**,[1] **and Yuefeng Ji**[1]

[1]*State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*School of Information Engineering, Henan University of Science and Technology, Luoyang 471023, China*

Correspondence should be addressed to Yuefeng Ji; jyf@bupt.edu.cn

Received 9 March 2021; Revised 20 May 2021; Accepted 19 July 2021; Published 16 August 2021

Academic Editor: Bo Rong

Mobile/multi-access edge computing (MEC) takes advantage of its proximity to end-users, which greatly reduces the transmission delay of task offloading compared to mobile cloud computing (MCC). Offloading computing tasks to edge servers with a certain amount of computing ability can also reduce the computing delay. Meanwhile, device-to-device (D2D) cooperation can help to process small-scale delay-sensitive tasks to further decrease the delay of tasks. But where to offload the computing tasks is a critical issue. In this article, we integrate MEC and D2D cooperation techniques to optimize the offloading decisions and resource allocation problem in D2D-enabled three-tier MEC networks for Internet of Things (IoT). Mobile devices (MDs), edge clouds, and central cloud data center (DC) make up these three-tier MEC networks. They cooperate with each other to finish the offloading tasks. Each task can be processed by MD itself or its neighboring MDs at device tier, by edge servers at edge tier, or by remote cloud servers at cloud tier. Under the maximum energy cost constraints, we formulate the cooperative offloading problem into a mixed-integer nonlinear problem aiming to minimize the total delay of tasks. We utilize the alternating direction method of multipliers (ADMM) to speed up the computing process. The proposed scheme decomposes the complicated problem into 3 smaller subproblems, which are solved in a parallel fashion. Finally, we compare our proposal with D2D and MEC networks in simulations. Numerical results validate that the proposed D2D-enabled MEC networks for IoT can significantly enhance the computing abilities and reduce the total delay of tasks.

1. Introduction

Mobile devices (MDs) have limited computing resources and power capacity due to their portable sizes [1]. To cope with computation-intensive, delay-sensitive, and high-energy-cost tasks, mobile/multi-access edge computing (MEC) has been proposed to deploy resource-rich servers at base stations (BSs) within the proximity of MDs [2–5]. In Internet of Things (IoT), MEC has been advocated as a promising technique for providing massive MDs with enhanced computing and storage capabilities [6].

For one thing, due to the physical proximity, MEC can significantly reduce the transmission latency [7, 8] involving in communication compared with mobile cloud computing (MCC) [9–12]. For another thing, MEC enables MDs to perform computation offloading, which can migrate their computing tasks [13] to resource-rich nodes and send results back to the MDs [14]. Offloading has always been a hot topic which can guarantee low computing time, as well as saving the battery energy of MDs [15], increasing the network throughput. Therefore, MEC can improve users' satisfaction [16] or quality-of-experience (QoE) of the end-users [17], relieve network congestion [18, 19], prolong the battery lifetime [20] of MDs, and further reduce the total latency distinctly. Thus, MEC is an efficient solution for IoT.

Nevertheless, offloading can obtain computing resources at the expense of extra transmission delay and associated energy consumption, due to the communication between

the MDs and edge servers over the wireless channels [21]. Offloading may be really beneficial to MDs with large amounts of computation and relatively small amounts of communication [22].

However, the edge servers associated with BSs have limited computing resources and can not always meet the requirements of the served MDs [23]. D2D cooperation can help to solve this problem, where the helpers (resource-rich MDs) can execute computing tasks from resource-limited MDs [24]. D2D-assisted cooperative offloading services play a complementary role in MEC, which is helpful when processing some small-scale tasks for energy-efficiency [25]. When a task has already been done well through D2D cooperation, accordingly, it can free up radio resources for other purposes and reduce the uplink delay of the task [26].

Therefore, in this paper, we propose cooperative computation offloading in D2D-enabled MEC networks for IoT aiming to minimize the total delay of tasks with the maximum energy consumption constraints. By leveraging the advantages of both MEC and D2D cooperation techniques, computing resources can be fully used. Our contributions can be summarized as follows:

(1) We propose a computing architecture where cooperative offloading can be carried out in the D2D-enabled three-tier MEC networks. Each task can be processed locally by MD itself or its neighboring MDs at device tier, by edge servers at edge tier, or by remote cloud servers at cloud tier

(2) The proposed three-tier MEC networks provide MDs with multiple optional offloading destinations. We formulate the offloading decisions and resource allocation in this MEC networks as an optimization problem aiming to minimize the total delay of tasks

(3) The formulated problem is a mixed-integer nonlinear problem which is hard to solve. We decompose the problem into 3 subproblems. Then, we solve it utilizing the alternating direction method of multipliers (ADMM). Extensive simulation results validate that the proposed scheme is effective

The organization of this article is as follows. Section 2 provides a review of related work. Sections 3 presents the system model, including the network model, communication model, and computing model. In Section 4, we formulate the problem and give the equivalent form. In Section 5, we propose a parallel optimization framework to solve the problem and develop an efficient computation offloading scheme. Related simulations are provided in Section 6. Finally, we conclude this article in Section 7.

2. Related Work

Extensive research has been conducted on MEC. Many existing works paid attention to the computation offloading problem [15, 16, 20, 21, 27–32]. Some works jointly considered the computation offloading policy and the involved resource allocation [15, 33–35]. Others jointly considered the offloading decisions, content caching [28, 36], and the resource allocation [20, 30]. All of these above works focused on two-tier edge computing networks.

Works focusing on the task offloading problem in three-tier edge computing networks can be found in [17, 37–40]. Tong et al. [39] first proposed a three-tier hierarchical edge cloud architecture to maximize the amount of peak mobile workloads from MDs being served. The performance of the proposed hierarchical edge cloud architecture was evaluated by a small-scale system experiment. The deployment of edge servers and cloud servers formed a three-tier MEC architecture, where local computing, edge computing, and cloud computing could coexist and cooperate to assist the task execution [38]. This scheme provided multiple offloading decisions for devices. The different offloading decisions may largely impact the network performance [21]. For example, offloading the tasks to the edge node or remote cloud DC will inevitably incur additional long communication delay [39], whereas executing the task locally may result in larger computing delay. Consequently, it is critical for devices to make proper offloading decisions in the three-tier computing networks.

Some works study cooperative offloading in MEC networks. du et al. [40] considered vertical cooperation between the fog and the cloud, in which the application can be processed in the MD locally, in the fog or in the cloud. The authors assumed that the offloading requests were usually very small; no buffer was needed for queueing the computing requests. Xiao and Krunz [17] designed optimal workload allocation solutions in a cooperative fog computing network. Instead of always relying on the cloud data center to process its unprocessed workload, each fog node can also forward part or all of its unprocessed workload to its neighboring fog nodes to further improve the QoE of its users. This paper solved the optimal workload allocation problem with the distributed ADMM. Wang et al. [38] developed a cooperative task offloading and computing resource allocation scheme in three-tier computing networks, considering the cooperation among the devices, the edge servers, and cloud servers vertically as well as the cooperation between the edge nodes horizontally. In these above works, they did not consider the cooperation among MDs.

D2D cooperation focuses on the collaborative computation among MDs. Feng et al. [24] developed a computation offloading scheme based on D2D communications, in which resource-limited MDs could offload their computation-intensive tasks to appropriate nearby resource-rich MDs. Xing et al. [41] studied D2D-enabled multihelper MEC networks, where a local user could be helped by its nearby wireless devices serving as helpers for cooperative computation. Literature [26] considered offloading computational tasks to nearby devices or to an edge cloud and developed a decentralized algorithm to allocate the computational tasks among them. However, these works were not carried out in three-tier networks.

In MEC networks, there are horizontal collaboration at the device tier [41] and the edge tier [17] as well as the vertical collaboration [40] among MDs, edge nodes, and cloud nodes [2]. Different from the above literatures, our paper

proposes cooperative computation offloading in D2D-enabled three-tier MEC networks for IoT. Each task can be processed locally by MD itself or its neighboring MDs, by edge servers at edge tier, or by cloud servers at cloud tier.

3. System Model

In this section, the system model adopted in this paper is described. We introduce the proposed network model and present the communication and the computing model in detail. Finally, we calculate the total delay of tasks and energy consumption.

3.1. Network Model. As depicted in Figure 1, we propose a D2D-enabled three-tier MEC network architecture as a part of the IoT, which consists of N MDs labeled as $\mathcal{N} = \{1, 2, \cdots, N\}$, M base stations (BSs), and a remote central cloud DC equipped with many remote cloud servers. Each MD has the ability of D2D communication and is connected to the closest BS via a wireless link. Each BS is equipped with some edge servers in edge cloud at edge tier and can be regarded as an edge node. The BSs are connected to the remote cloud servers in central cloud DC at cloud tier via a low-latency wired backhaul link (e.g., optical link) [4]. Optical backhauling is a promising solution [42, 43] to offer high throughput, low latency, and reduced energy consumption. Moreover, the flexibility and reliability can be effectively improved by the adoption of artificial intelligence-driven autonomous optical networks [44].

In this D2D-enabled cooperative architecture, we assume that each MD has a computing task to be processed locally by MD itself or its neighboring MDs at device tier, by edge cloud, or by central cloud DC. MDs can help each other when they are free or have surplus computing resources. The helper may be rewarded in terms of accumulated reward points. MDs with more accumulated reward points can be helped with higher precedence when they need help. Multiple neighboring local MDs can communicate with each other through WiFi [45] or Bluetooth [46]. The network operator (e.g., BS) can gather sufficient D2D connectivity information through network-assisted device discovery and local information reporting by the devices [25]. If the network condition is bad, the task of each local MD is small and/or delay-sensitive; MDs may be more willing to choose D2D cooperation due to short transmission time. That is to say, these tasks prefer to be processed among local MDs.

The computation offloading process is shown in Figure 2 [35]. Each MD can send an offloading request including the information of the MD such as its local processing capability and power consumption, the properties of the task such as the size and emergency of the task, and the maximum tolerable delay [40]. According to this collected information, MDs finally decide where should the tasks be processed. The computing task of MD n can be completed locally within its CPU, or by its neighboring MD via D2D cooperation, or remotely in the edge cloud via MEC offloading, or further up the cloud DC via cloud offloading [35].

After being processed in one of these nodes, the result of the computing task is transmitted back to the corresponding

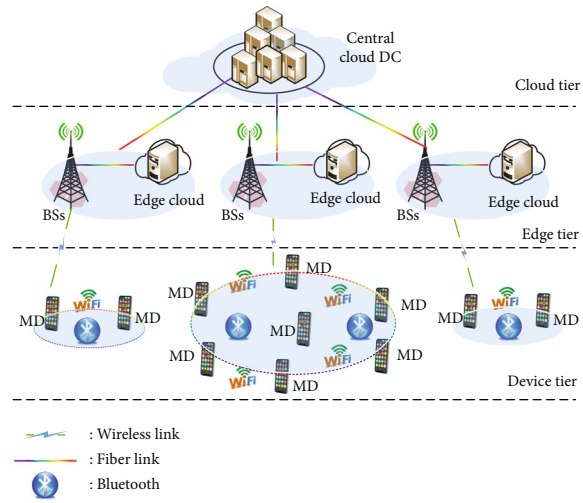

Figure 1: A D2D-enabled three-tier MEC network.

MD. Similar to many previous works about MEC [18, 20, 21, 26], we neglect the return result time of the computed tasks in this work. The assumption is justified for many applications including face recognition and anomaly detection, where the size of the result is much smaller than the size of the uplink data. We do not consider the queueing of offloading tasks, as in [19, 40]. To obtain tractable analysis, we also omit the delay due to decision-making [20].

3.2. Communication Model. For the communication model, the computing task of MD n is described by $\Gamma_n = (L_n, \tau_n^{\max}, C_n)$, where L_n (in bits) stands for the input data size of task Γ_n, $\tau_n^{\max}$ (in s) stands for the task computation deadline for MD n, and C_n (in CPU cycles) stands for the number of required CPU cycles to accomplish the task Γ_n.

When a task of an MD can not be processed well by MD itself or its neighboring MDs, the task can be sent to the BS. The bandwidth resource of the BS is denoted as B Hz. However, the limited radio bandwidth needs to be allocated among all the edge-processing and the cloud-executing MDs for communication with the edge node. Within our communication model, all the devices equally share the bandwidth. The uplink bandwidth of MD n is B_n Hz; the uplink achievable transmission rate R_n for radio access can be approximated by Shannon's formula:

$$R_n = B_n \cdot \log_2\left(1 + \frac{p_n B_n G_n}{\sigma^2}\right), \tag{1}$$

where p_n is the transmission power density of MD n, G_n stands for the channel gain between the MD n and the BS, and σ^2 denotes the power of additive white Gaussian noise.

Furthermore, the uplink transmission delay for MD n offloading the task Γ_n to the BS is calculated as

$$T_n^{tr} = \frac{L_n}{R_n}. \tag{2}$$

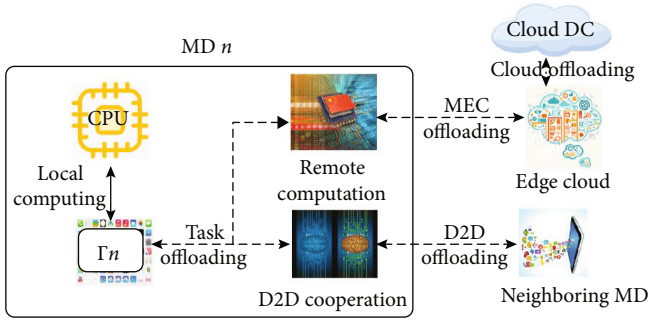

Figure 2: Computation offloading process.

And the transmission energy consumption for MD n offloading the task Γ_n to the BS can be given as

$$E_n^{tr} = P_n T_n^{tr}, \tag{3}$$

where P_n is the total power consumption of MD n for transmission in the uplink, consisting of the static power consumption and transmit power [38].

3.3. Computing Model. We denote the offloading decisions for MD n by $x_{n,j}, y_n, z_n \in \{0, 1\}$, where $x_{n,j} = 1, y_n = 1, z_n = 1$ indicate that the task of MD n is executed by MD itself (e.g., $n = j$) or its neighboring MD j, by edge server, or by cloud server, respectively; otherwise, $x_{n,j} = 0, y_n = 0, z_n = 0$. Thus, the offloading decisions of device n is constrained by

$$\sum_{j=1}^{N} x_{n,j} + y_n + z_n = 1, \quad \forall n \in \mathcal{N}. \tag{4}$$

In the constraint, it should be noted that one and only one of the offloading decisions $x_{n,j}$, y_n, and z_n for the MD n can be 1 at any time. It also implies that the task of each MD is impartible in this paper.

Next, we will discuss the computation overhead in terms of processing delay and energy consumption of MD n with different offloading approaches.

3.3.1. Local Processing. For the local task processing approach, let f_n^L denote the local computing capability (in CPU cycles/s) of MD n. Then, the computing delay of processing the task Γ_n locally can be

$$T_n^L = T_{\Gamma_n}^n = \frac{C_n}{f_n^L}. \tag{5}$$

The CPU power consumption is a function of execution frequency f_n^L. We can compute the energy consumption of MD n as [47, 48]

$$E_n^L = 10^{-11} C_n \left(f_n^L\right)^2. \tag{6}$$

From (5) and (6), we can notice that both T_n^L and E_n^L are only related to C_n and f_n^L, which are the known inherent features of MD n and its computing task.

3.3.2. D2D Cooperation. For D2D cooperation, we define $d_{n,j}$ as the D2D link between local MD n and j, P_n^D as the power consumption of MD n, and $T_{n,j}^{tr}$ as the round trip time (RTT) for task transferring between the D2D pairs. This RTT is assumed to be stable during an offloading period, while it may change across different periods. It can be estimated at the beginning of the offloading period, by using the history information or empirical formula [49]. Thus, when the task Γ_n is executed finally at MD j, the total delay T_n^D consists of the communication delay $T_{n,j}^{tr}$ between MD n and MD j and the processing delay $T_{\Gamma_n}^j$ at MD j. It can be calculated as

$$T_n^D = T_{\Gamma_n}^j + T_{n,j}^{tr}, \quad \forall j \in \mathcal{N}, \tag{7}$$

according to (5), $T_{\Gamma_n}^j = C_n / f_j^L$. Note that $T_{n,j}^{tr} = 0$ when $n = j$, because there is no task to transfer within the local MD. Thus, the delays of tasks in the device tier all meet (7).

In this case, the energy cost of MD n is mainly the transmission energy consumption. It can be given as

$$E_n^D = P_n^D T_{n,j}^{tr}. \tag{8}$$

3.3.3. Edge Computing. For the edge computing approach, the total computing capacity of the BS is denoted as F (in CPU cycles/s), which can be allocated to the computing tasks processed by the BS. Denote f_n^B as the computing resource allocated to MD n by the BS. Thus, the computing delay of MD n executed by BS is given as

$$T_n^B = \frac{C_n}{f_n^B}. \tag{9}$$

When the tasks are computed in the edge server, the MD n sends the task Γ_n to the associated BS. Then, the BS will process the offloaded task by employing its computing

resources. Combining with equation (2), the total delay of task Γ_n processed by the BS is calculated by

$$T_n^E = T_n^{tr} + T_n^B. \tag{10}$$

3.3.4. Cloud Computing. For the computing task offloaded to the cloud DC, MD n firstly transmits the task Γ_n to the nearest BS through a wireless link. Then, the BS forwards the task Γ_n to the cloud DC through a fiber optical link. Since the cloud servers have plenty of computing resources and the fiber optical link between the edge cloud and the cloud DC is of sufficiently large capacity, the allocation of these resources will not be discussed here. Ordinarily, the computing capability of the cloud DC is much higher than that of the MEC servers.

Let the RTT for task transmission between the edge server at BS and cloud server be $T_{B,C}$. Denote f_n^C as the cloud computing capability (in CPU cycles/s) assigned to MD n, which is assumed to be a predetermined value according to the cloud computing service [40]. Therefore, the cloud computing delay of MD n is given by

$$T_n^{fC} = \frac{C_n}{f_n^C}. \tag{11}$$

Then, combining with equation (2), the total delay of cloud processing for MD n can be expressed as

$$T_n^C = T_n^{tr} + T_{B,C} + T_n^{fC}. \tag{12}$$

The above notations that will be used in this paper are summarized in Table 1.

As mentioned earlier, we ignore the return result time of the computed tasks. According to (2)–(12), the total task delay and energy consumption of MD n are calculated, respectively, as

$$T_n = \sum_{j=1}^{N} x_{n,j} T_n^D + y_n T_n^E + z_n T_n^C, \tag{13}$$

$$E_n = x_{n,n} E_n^L + \sum_{j \in N \setminus \{n\}} x_{n,j} E_n^D + \left(1 - \sum_{j=1}^{N} x_{n,j}\right) E_n^{tr}. \tag{14}$$

4. Problem Formulation and Transformation

4.1. Problem Formulation. To minimize the total delay, we formulate the task offloading and resource allocation problem with the maximum energy cost constraints. Let the offloading strategy vector of MD n as $\Theta_n = \{x_{n,1}, x_{n,2}, \cdots, x_{n,N}, y_n, z_n\}$ and the offloading decision profile of all devices as $\Theta = \{\Theta_n, n \in \mathcal{N}\}$. The computing resource allocation vector of all devices is denoted as $\Omega = \{f_n^B, n \in \mathcal{N}\}$. The maximum energy cost of MD n is denoted as $E_n^{\max}$ (e.g., the battery capacity). As a result, the joint task offloading and resource allocation problem are formulated as

$$P1: \min_{\Theta,\Omega} \quad \sum_{n=1}^{N} T_n \tag{15}$$

$$\text{s.t.} \quad \sum_{j=1}^{N} x_{n,j} + y_n + z_n = 1, \quad \forall n \in \mathcal{N}, \tag{16}$$

$$x_{n,j}, y_n, z_n \in \{0, 1\}, \quad \forall n \in \mathcal{N}, \tag{17}$$

$$\sum_{n=1}^{N} f_n^B \le F, \tag{18}$$

$$x_{n,n} E_n^L + \sum_{j \in \mathcal{N} \setminus \{n\}} x_{n,j} E_n^D + \left(1 - \sum_{j=1}^{N} x_{n,j}\right) E_n^{tr} \le E_n^{\max}. \tag{19}$$

The constraints (16) and (17) denote that each task is processed locally by MD itself or by its neighboring MDs, by edge servers, or by cloud servers. The constraint (18) states the computing resource consumption of all the MDs can not exceed the computing capacity of the edge servers. The constraint (19) indicates the maximum energy cost of MD n. Based on (7), (10), (12), and (13), the total delay of MD n can be written as

$$\begin{aligned} T_n &= \sum_{j=1}^{N} x_{n,j} T_n^D + y_n T_n^E + z_n T_n^C \\ &= \sum_{j=1}^{N} x_{n,j} \left(T_{\Gamma_n}^j + T_{n,j}^{tr}\right) + y_n \left(T_n^{tr} + T_n^B\right) \\ &\quad + z_n \left(T_n^{tr} + T_{B,C} + T_n^{fC}\right). \end{aligned} \tag{20}$$

The optimization problem $P1$ is a mixed-integer nonlinear problem and is difficult to solve due to the following observation. Since that $x_{n,j}$, y_n, and z_n are binary variables, the feasible set of problem (15) is not convex, so that the objective function of problem (15) is not convex. As is shown, problem (15) is a mixed discrete and nonconvex optimization problem, and this kind of problem is usually considered as NP-hard [20]. Then, we rewrite the objective function (15) as

$$\begin{aligned} \Psi(\Theta) = \sum_{n=1}^{N} T_n &= \sum_{n=1}^{N} \sum_{j=1}^{N} x_{n,j} \left(T_{\Gamma_n}^j + T_{n,j}^{tr}\right) + \sum_{n=1}^{N} y_n \left(T_n^{tr} + T_n^B\right) \\ &\quad + \sum_{n=1}^{N} z_n \left(T_n^{tr} + T_{B,C} + T_n^{fC}\right). \end{aligned} \tag{21}$$

In that, $P1$ is a convex problem with all binary variables fixed. We can use the branch-and-bound method to find a near-optimal solution [38]. But as far as our objective

Table 1: Notation and corresponding definition.

Notation	Definition
Γ_n	Computing task of MD n
L_n	Data size of task Γ_n (in bits)
C_n	CPU cycles required to accomplish the task Γ_n
B	The bandwidth of the BS
B_n	The uplink bandwidth of MD n
R_n	The uplink transmission rate of MD n
p_n	The transmission power density of MD n
P_n^D	The power consumption of MD n in D2D cooperation
P_n	The total power consumption of MD n
G_n	The channel gain between the MD n and the BS
σ^2	The power of the additive white Gaussian noise
f_n^L	The local computing capability of MD n (in CPU cycles/s)
f_n^B	The computing resource allocated to MD n by the BS
f_n^C	The computing resource allocated to MD n by the cloud
F	The computing capacity of the BS
$\tau_d^{\max}$	The task computation deadline for MD n
T_n^{tr}	The uplink transmission delay of task Γ_n offloaded to the BS
$T_{\Gamma_n}^j$	The computing delay of task Γ_n processed at MD j
$T_{n,j}^{tr}$	The average round trip time between MD n and j
T_n^L	The delay of task Γ_n processed locally
T_n^D	The total delay of task Γ_n via D2D cooperation
T_n^B	The computing delay of MD n by the BS
$T_{B,C}$	The round trip time between BS and the cloud servers
T_n^{fC}	The cloud computing delay of MD n
T_n^C	The total delay of task Γ_n processed by the cloud servers
E_n^{tr}	The transmission energy consumption of task Γ_n to the BS
E_n^L	The energy consumption of processing the task Γ_n locally
E_n^D	The energy consumption of processing Γ_n via D2D cooperation

function (15) is concerned, we should note that the complexity of the method remains exponential. The scale of the problem becomes large as the number of MDs grows, which makes it hard to obtain the solution in a reasonable amount of time. Then, in the next section, we will decouple the optimization problem (15) to present an effective computation offloading scheme ADMM to solve the problem $P1$.

4.2. Problem Transformation. To make it possible for each offloading decisions participating in the computation, we will decompose the problem $P1$ into several small subproblems, so that it can be solved in a distributed manner. Thus, we need to properly handle the coupling constraint among variables. Specially speaking, the optimization variables $x_{n,j}$, y_n, and z_n are coupled in the constraints (16), which make the problem inseparable. Thus, to make the problem separable, we introduce the local copies of the global variables and denote them as $\widehat{x}_{n,j}$, $\widehat{y}_n$, and $\widehat{z}_n$, respectively. Then, we have

$$\begin{cases} \widehat{x}_{n,j} = x_{n,j}, & \forall n \in \mathcal{N}, j \in \mathcal{N}, \\ \widehat{y}_n = y_n, & \forall n \in \mathcal{N}, \\ \widehat{z}_n = z_n, & \forall n \in \mathcal{N}. \end{cases} \tag{22}$$

Based on equation (22), the coupling constraint of (16) can be written as

$$\sum_{j=1}^{N} \widehat{x}_{n,j} + \widehat{y}_n + \widehat{z}_n = 1, \quad \forall n \in \mathcal{N}. \tag{23}$$

Let $\widehat{\Theta} = \{\widehat{x}_{n,1}, \widehat{x}_{n,2}, \cdots, \widehat{x}_{n,N}, \widehat{y}_n, \widehat{z}_n, n \in \mathcal{N}\}$ be the local variables vector of all MDs. Then, we give the equivalent global consensus version of $P1$ by substituting (16) with (22) and (23) shown as

$$\begin{aligned} P2: \min_{\Theta,\Omega,\widehat{\Theta}} \quad & \Psi(\Theta) \\ \text{s.t.} \quad & (17), (18), (19), (22), (23). \end{aligned} \tag{24}$$

Now, it is obvious that in problem $P2$, the objective functions $\Psi(\Theta)$ with feasible sets are separable. But the consensus constraints (23) remain coupled. As far as the problem $P2$ is concerned, the numbers of variables and constraints reach $N^2 + 2N$ and $2N^2 + 4N$, respectively. The size of this problem becomes extremely large as the number of MDs grows.

ADMM is a promising approach to solve the large-scale optimization problem [20], which is originally devised to solve convex optimization problems. Recently, it has been further explored to solve nonconvex problems due to the flexibility of the ADMM framework [30]. In the next section, we will apply ADMM to solve the problem in a distributed fashion.

5. Efficient Computation Offloading Scheme

In this section, we develop a parallel optimization framework based on ADMM and difference of convex function (D.C.) programming. First, we derive the augmented Lagrangian function. With corresponding global consensus constraints, we formulate the ADMM iteration steps [28, 38, 50]. Then, the update methods for ADMM iterations are presented, and the overall algorithm is summarized.

5.1. Augmented Lagrangian and ADMM Variables. According to [50], the augmented Lagrangian function is expressed as

$$
\begin{aligned}
&L_\rho\left(\Theta, \widehat{\Theta}, \lambda, \mu, \gamma\right) \\
&= \varphi(\Theta) + \sum_{n=1}^{N}\sum_{j=1}^{N}\lambda_{n,j}\left(x_{n,j} - \widehat{x}_{n,j}\right) + \sum_{n=1}^{N}\mu_n\left(y_n - \widehat{y}_n\right) \\
&\quad + \sum_{n=1}^{N}\gamma_n\left(z_n - \widehat{z}_n\right) + \frac{\rho}{2}\sum_{n=1}^{N}\sum_{j=1}^{N}\left(x_{n,j} - \widehat{x}_{n,j}\right)^2 \\
&\quad + \frac{\rho}{2}\sum_{n=1}^{N}\left(y_n - \widehat{y}_n\right)^2 + \frac{\rho}{2}\sum_{n=1}^{N}\left(z_n - \widehat{z}_n\right)^2,
\end{aligned} \tag{25}
$$

where $\lambda = \{\lambda_{n,j}\}_{n,j\in\mathcal{N}}$, $\mu = \{\mu_n\}_{n\in\mathcal{N}}$, and $\gamma = \{\gamma_n\}_{n\in\mathcal{N}}$ are the Lagrange multipliers with respect to the constraint (23), respectively. $\rho > 0$ is the so-called penalty parameter, augmented Lagrangian parameter, which is a constant parameter intend to adjust the convergence speed of ADMM [50].

With the ADMM method to solve the problem $P2$, the resulting optimization steps are described as follows.

5.2. Global Variable Update. The global variables $\{\Theta, \lambda, \mu, \gamma\}$ are updated in the $[t+1]$-th iteration by settling the following optimization problem [51]:

$$
\begin{aligned}
P3: \min_{\Theta,\Omega} \quad & L_\rho\left(\Theta, \widehat{\Theta}^{[t]}, \lambda^{[t]}, \mu^{[t]}, \gamma^{[t]}\right) \\
\text{s.t.} \quad & (17), (18), (19),
\end{aligned} \tag{26}
$$

where the superscript $[t]$ stands for the iteration index.

We denote $\mathrm{x}_n = \{x_{n,j}, j \in \mathcal{N}\}$ as the local offloading decision vector of MD n, $\mathrm{y} = \{y_n, n \in \mathcal{N}\}$ as the decision vector of offloading tasks to BS j. For cloud computing, $\mathrm{z} = \{z_n, n \in \mathcal{N}\}$ is the decision vector of offloading tasks to the cloud DC. Then, omitting the local copies of the variables $\widehat{x}_{n,j}$, $\widehat{y}_n$, $\widehat{z}_n$, the objective function in $P3$ can be rewritten as

$$
L_\rho\left(\Theta, \widehat{\Theta}^{[t]}, \lambda^{[t]}, \mu^{[t]}, \gamma^{[t]}\right) = \sum_{j=1}^{M} f_j(x_n) + g(y) + h(z), \tag{27}
$$

where the functions $f_j(x_n)$, $g(y)$, and $h(\mathrm{z})$ are given

$$
f_j(x_n) = \sum_{n\in\mathcal{N}}\left[\frac{\rho}{2}\left(x_{n,j} - \widehat{x}_{n,j}^{[t]}\right)^2 + \left(T_n^L + \lambda_n^{[t]}\right)x_{n,j}\right], \tag{28}
$$

$$
g(y) = \sum_{n\in\mathcal{N}}\left[\frac{\rho}{2}\left(y_n - \widehat{y}_n^{[t]}\right)^2 + \left(T_n^E + \mu_n^{[t]}\right)y_n\right], \tag{29}
$$

$$
h(z) = \sum_{n\in\mathcal{N}}\left[\frac{\rho}{2}\left(z_n - \widehat{z}_n^{[t]}\right)^2 + \left(T_n^C + \gamma_n^{[t]}\right)z_n\right]. \tag{30}
$$

Therefore, it is easy to recognize that the objective function and the feasible region in $P3$ are separable completely. As a result, $P3$ can be decomposed equivalently into the following three subproblems ($P1'$, $P2'$, and $P3'$), which are given by

$$
\begin{aligned}
P1': \min_{x_n} \quad & f_j(x_n) \\
\text{s.t.} \quad & (18), (19), x_{n,j} \in \{0,1\}, \quad \forall n \in \mathcal{N},
\end{aligned} \tag{31}
$$

$$
\begin{aligned}
P2': \min_{y} \quad & g(y) \\
\text{s.t.} \quad & y_n \in \{0,1\}, \quad \forall n \in \mathcal{N}.
\end{aligned} \tag{32}
$$

Meanwhile, the subproblem $P3'$ related to the cloud servers is written as

$$
\begin{aligned}
P3': \min_{z} \quad & h(z) \\
\text{s.t.} \quad & z_n \in \{0,1\}, \quad \forall n \in \mathcal{N}.
\end{aligned} \tag{33}
$$

5.2.1. The Solution of the First Subproblem $P1'$. There are some difficulties in solving this subproblem $P1'$ with the existence of the binary variable constraint. To overcome the difficulties, an equivalent transformation of the original problem is necessary. We can transform the binary constraints in $P1'$ as

$$
\sum_{n\in\mathcal{N}}\left(x_{n,j} - x_{n,j}^2\right) \le 0, \tag{34}
$$

$$
0 \le x_{n,j} \le 1, \quad \forall n \in \mathcal{N}. \tag{35}
$$

By replacing the binary constraint with (34) and (35), problem $P1'$ is rewritten as

$$
\begin{aligned}
P4: \min_{\mathrm{x}_n} \quad & f_j(x_n) \\
\text{s.t.} \quad & (18), (19), (34), (35), \quad \forall n \in \mathcal{N}.
\end{aligned} \tag{36}
$$

After the above equivalent transformations, the objective function of $P4$ is linear and constraints except (34) are convex. To deal with (34), we further transform $P4$ as given by

$$
\begin{aligned}
P5: \min_{x_j} \quad & f_j\left(x_j\right) + \alpha\sum_{n\in\mathcal{N}}\left(x_{n,j} - x_{n,j}^2\right) \\
\text{s.t.} \quad & (18), (19), (35), \quad \forall n \in \mathcal{N},
\end{aligned} \tag{37}
$$

where α acts as a penalty factor. It is proven for a sufficiently large amount; the problem $P5$ is equivalent to $P4$ [52].

Here, we define $\varphi_j(\mathrm{x}_n)$ and $\phi_j(x_{n,j})$ as

$$
\varphi_j(x_n) = f_j(x_n) + \alpha\sum_{n\in\mathcal{N}}x_{n,j}, \tag{38}
$$

$$
\phi_j\left(x_{n,j}\right) = \alpha\sum_{n\in\mathcal{N}}x_{n,j}^2. \tag{39}
$$

Then, it can be found that the objective function can be written as the difference of the two convex functions $\varphi_j(x_n)$ and $\phi_j(x_{n,j})$. We can find that $P5$ is a D.C. programming problem [53]. Therefore, to solve $P5$, we can exploit the sequential convex approximation [24] of $\phi_j(x_{n,j})$ in the x_n

domain. $\phi_j(x_{n,j})$ can be approximated by its first-order Taylor expansion, and the objective function can be derived as

$$\varphi_j(x_n) - \phi_j\left(x_{n,j}\right) + \nabla_{x_j}\phi_j\left(x_{n,j}\right) \times \left(x_{n,j} - x_{n,j}^{([t])}\right), \tag{40}$$

where $\nabla_{x_n}\phi_j(x_{n,j})$ is the gradient of $\phi_j(x_{n,j})$ at x_n and can be expressed as $\nabla_{x_n}\phi_j(x_{n,j}) = (\partial\phi_j(x_{n,j}))/\partial x_{n,j}, \forall j \in \mathcal{N}$.

Then, we can obtain an approximation solution to get a local optimal solution $x_n^{[t]}$ during each iteration, which is sketched in Algorithm 1, where $x_n^{[t]}$ is the solution at the t -th iteration. As a result, $P5$ is a convex problem and can be solved efficiently by the standard convex optimization methods [54]. We can obtain near-optimal solution $x_n^{[t]}$ of P 5 by iteratively solving.

It is worth noting that the solution $x_n^{[t]}$ is better than the previous solution $x_n^{[t-1]}$. Since the constraint set is compact, the sequence $\{x_n^{[t]}\}_{t=1,2,3,\cdots,n}$ is proved to be convergent [52] by Cauchy theorem. The iterative process can stop after finite iterations. Given an initial point, due to the convexity of each iteration, Algorithm 1 converges to the same solution. In the issue, the convergence properties in Algorithm 1 could facilitate the convergence of ADMM.

5.2.2. The Solutions of the Subproblem P2′. For the subproblem $P2'$, it is an unconstrained binary optimization problem. Due to the fact that possible values of y_n are 0 and 1, we have $y_n^2 = y_n$. Then, the expression after the summation sign of (29) is

$$\frac{\rho}{2}\left(y_n - 2y_n\widehat{y}_n^{[t]^2} + \widehat{y}_n^{[t]^2}\right) + \left(T_n^E + \mu_n^{[t]}\right)y_n. \tag{41}$$

By comparing the corresponding objective function values of two possible solutions, the solution can be given by

$$y_n = \begin{cases} 0 & \frac{\rho}{2} - \rho\widehat{y}_n^{[t]} + T_n^E + \mu_n^{[t]} > 0, \\ 1 & \frac{\rho}{2} - \rho\widehat{y}_n^{[t]} + T_n^E + \mu_n^{[t]} \le 0. \end{cases} \tag{42}$$

5.2.3. The Solutions of the Subproblem P3′. For the subproblem $P3'$, it is also an unconstrained binary optimization problem as $P2'$. Thus, the solution can also be given as

$$z_n = \begin{cases} 0 & \frac{\rho}{2} - \rho\widehat{z}_n^{[t]} + T_n^C + \gamma_n^{[t]} > 0, \\ 1 & \frac{\rho}{2} - \rho\widehat{z}_n^{[t]} + T_n^C + \gamma_n^{[t]} \le 0. \end{cases} \tag{43}$$

5.3. Local Variable Update. Now, we move on to the local variables. After obtaining the global variables, at the $[t+1]$-th iteration, the local variables $\widehat{\Theta}$ are updated by solving the following optimization problem:

1. Initialization:
Choose an initial feasible solution $\{x_{n,j}^{(0)}\}$ of $P1'$,
t=0.
2. Iteration:
repeat
Solve the following convex problem $P5$:
$\min_{x_n} \varphi_j(x_n) - \phi_j(x_{n,j}) + \nabla_{x_n}\phi_j(x_{n,j}) \times (x_{n,j} - x_{n,j}^{([t])})$
s.t. (18),(19),(35)
obtain the optimal solution $x_n^{[t+1]}$
update t=t+1
3. until convergence of $\mathbf{x}_n$.

Algorithm 1: D.C-based algorithm for subproblem $P1'$.

$$\begin{aligned} P6: \min_{\widehat{\Theta}} \quad & L_\rho\left(\Theta^{[t+1]}, \widehat{\Theta}^{[t]}, \lambda^{[t]}, \mu^{[t]}, \gamma^{[t]}\right) \\ \text{s.t.} \quad & (23), \widehat{x}_{n,j}, \widehat{y}_n, \widehat{z}_n \in \{0,1\}, \quad \forall n \in \mathcal{N}, j \in \mathcal{N}. \end{aligned} \tag{44}$$

After eliminating some constant terms from the objective function, it is equivalent to solve the following problem:

$$\begin{aligned} P7: \min_{\widehat{\Theta}} \quad & \sum_{n=1}^{N} F\left(\widehat{\Theta}_n\right) \\ \text{s.t.} \quad & (23), \widehat{x}_{n,j}, \widehat{y}_n, \widehat{z}_n \in \{0,1\}, \quad \forall n \in \mathcal{N}, j \in \mathcal{N}, \end{aligned} \tag{45}$$

where the function $F(\widehat{\Theta}_n)$ is given by

$$\begin{aligned} F\left(\widehat{\Theta}_n\right) = & \sum_{j=1}^{M}\left[\frac{\rho}{2}\widehat{x}_{n,j}^2 - \left(\rho x_{n,j}^{[t+1]} + \lambda_{n,j}^{[t]}\right)\widehat{x}_{n,j}\right] \\ & + \left[\frac{\rho}{2}\widehat{y}_n^2 - \left(\rho y_n^{[t+1]} + \mu_n^{[t]}\right)\widehat{y}_n\right] \\ & + \left[\frac{\rho}{2}\widehat{z}_n^2 - \left(\rho z_n^{[t+1]} + \gamma_n^{[t]}\right)\widehat{z}_n\right]. \end{aligned} \tag{46}$$

As mentioned earlier, the constraint (23) implies that only one of $x_{n,j}$, y_n, and z_n for the MD n can be 1 at any time. Thus, there are total $N+2$ possible solutions for each MD n. We only need to calculate the corresponding objective function values of the $N+2$ different offloading decisions and choose the minimum one, which corresponds to the optimal solution. Formally, the optimal solution of P6 is written as

$$\begin{cases} \widehat{x}_{n,j} = 1, \widehat{y}_n = 0, \widehat{z}_n = 0, & \text{if } \mathrm{c}_{n,j} = C_{n,\min}, \forall n \in \mathcal{N}, \\ \widehat{x}_{n,j} = 0, \widehat{y}_n = 1, \widehat{z}_n = 0, & \text{if } \mathrm{c}_{n,N+1} = C_{n,\min}, \forall n \in \mathcal{N}, \\ \widehat{x}_{n,j} = 0, \widehat{y}_n = 0, \widehat{z}_n = 1, & \text{if } \mathrm{c}_{n,N+2} = C_{n,\min}, \forall n \in \mathcal{N}, \end{cases} \tag{47}$$

where $\mathrm{c}_{n,j} = (\rho/2) - \rho x_{n,j}^{[t+1]} - \lambda_{n,j}^{[t]}$, $\mathrm{c}_{n,N+1} = (\rho/2) - \rho y_n^{[t+1]} - \mu_n^{[t]}$, and $\mathrm{c}_{n,N+2} = (\rho/2) - \rho z_n^{[t+1]} - \gamma_n^{[t]}$ and $C_{n,\min}$ is the minimum

1. Initialization:
t =1, $\varepsilon = 0.01$, $\rho = 0.1$, $\lambda^{[t]} = 0$, and $t_{\max} = 2000$
Choose an initial feasible solution $\widehat{\Theta}^{[t]}$ satisfying (23)
2. Iteration:
while $\|\Theta - \widehat{\Theta}\|_2 \geq \varepsilon$ and $t \leq t_{\max}$ **do**
I) Global variable update: M computation units solve $P1'$.
and $P2'$ in parallel and a computation unit solves $P3'$ to
update the global variables $\{\Theta\}^{[t+1]}$
II) Local variable update: Update the local variables $\widehat{\Theta}^{[t+1]}$ based on (47)
III) Dual variable update: Update the dual variables $\{\lambda, \mu, \gamma\}^{[t+1]}$ based on the equations (48)-(50)
end while
3. Output:
Output the optimal solution $\{x_{n,j}, y_n, z_n\}^*$.

Algorithm 2: Parallel optimization algorithm via ADMM.

value of all possible solutions for MD n, which is expressed as $C_{n,\min} = \min\{c_{n,j\in N}, c_{n,N+1}, c_{n,N+2}\}$.

5.4. Dual Variable Update. After obtaining the global and local variables, we perform the process of dual variable update. In the $[t+1]$-th iteration, the dual variables are updated as follows:

$$\lambda_{n,j}^{[t+1]} = \lambda_{n,j}^{[t]} + \rho\left(x_{n,j}^{[t+1]} - \widehat{x}_{n,j}^{[t+1]}\right), \quad \forall n \in \mathcal{N}, j \in \mathcal{N}, \tag{48}$$

$$\mu_n^{[t+1]} = \mu_n^{[t]} + \rho\left(y_n^{[t+1]} - \widehat{y}_n^{[t+1]}\right), \quad \forall n \in \mathcal{N}, \tag{49}$$

$$\gamma_n^{[t+1]} = \gamma_n^{[t]} + \rho\left(z_n^{[t+1]} - \widehat{z}_n^{[t+1]}\right), \quad \forall n \in \mathcal{N}. \tag{50}$$

In summary, the efficient offloading scheme is obtained by sequential iteration for global variables, local variables, and dual variables. By leveraging the ADMM algorithm to the problem P3, we firstly minimize the augmented Lagrangian function (25) over the global variables and decompose the optimization problem into some smaller subproblems, which is executed in parallel to improve the computing speed. Then, we performed the local variable update based on equation (47). Finally, the dual variables are updated based on equations (48)–(50). The algorithm is summarized in Algorithm 2.

6. Evaluation

In this section, we will present simulation results to verify the performance of the proposed D2D-enabled three-tier MEC networks. In our simulations, the number of BSs is 4. Each BS has a radius of 400 m. And there are 80 MDs uniformly distributed in the coverage of each BS. We assume that each MD can establish one wireless link with the BS and one D2D link with a neighboring MD. The maximum range of each D2D link is set as 50 m. The channel gain model of all wireless links is chosen as $h = 127 + 30\log_{10} d$ (d in km), as suggested in [55]. Each MD has a task to be computed. The input data size of task n is $L_n \in [0.1, 4.0]$ Mbits. And the required computation amount of each task is assumed to be $C_n \in [2 \times 10^9, 5 \times 10^9]$ CPU cycles. The maximum delay tolerance of each task is 1 s. The computing capability of each MD is uniformly distributed from 0.5×10^9 to 2×10^9 CPU cycles/s. The maximum transmission power is set as 0.5 W [56] for each MD.

Table 2: Simulation parameters.

Parameters	Value
The bandwidth resource B_n	20 MHz [20]
The number of BSs M	4
The transmission power of MDs P_n	20 dBm [4]
Power of background noise σ^2	-174 dBm/Hz [57]
Path-loss model	$127 + 30 * \log_{10} d$, d (km)
Local computational capability f_n^L	0.5GHz [56]
Local computation power P_n^L	24 dBm [20]
The computing capability of BSs F	20 GHz [38]
The computing capability of cloud F	100 GHz [33]

All random variables are independent for different MDs, modeling heterogeneous mobile computing resources. The main system parameters used in the simulation are summarized in Table 2, if not specified.

6.1. Performance Comparison. We first compare the performance of the proposed D2D-enabled three-tier MEC framework with the other two networks [23, 35, 58]:

(1) In the D2D networks, tasks which can be cooperatively processed only among local MDs

(2) In the MEC networks, vertical cooperation among local MD itself, edge node, and remote cloud node.

(3) In the D2D-enabled MEC networks, where each task can be processed locally by MD itself or its neighboring MDs, by edge cloud, or by remote cloud.

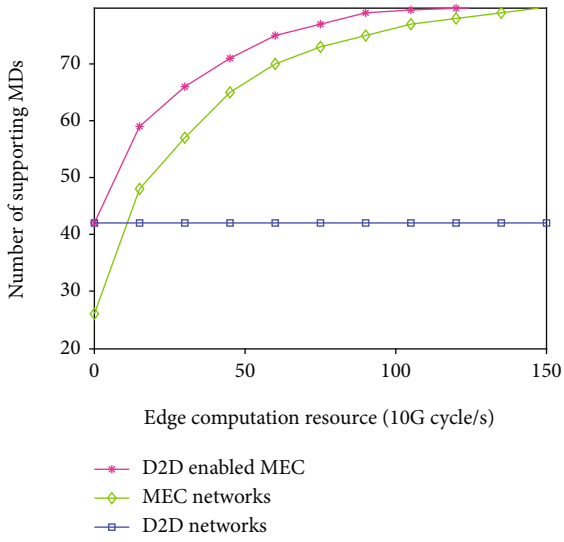

Figure 3: The number of supported MDs.

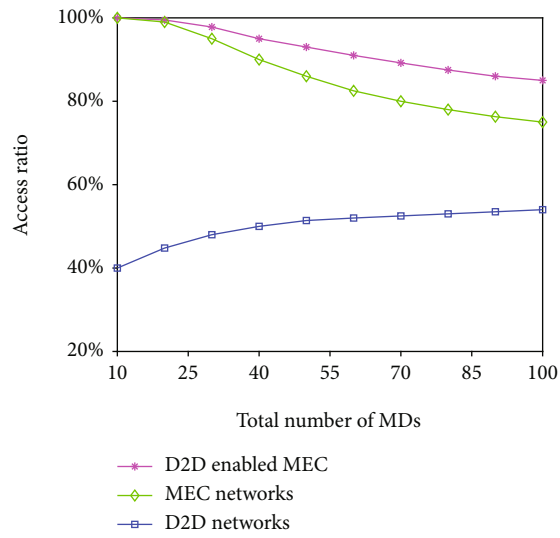

Figure 4: Access rate with different numbers of MDs.

From Figure 3, we can observe that the number of supporting MDs in these different networks to edge computing resources. First of all, the numbers of supporting MDs in the MEC networks and the D2D-enabled MEC networks both increases with the edge computing resources. While the number of supporting MDs in the D2D networks keeps invariable. The reason is that the D2D networks does not utilize the edge computing resources. Secondly, the D2D-enabled three-tier MEC networks achieves the best performance among these three systems. The reason is that the D2D-enabled MEC networks can utilize the computing resources of MDs, the edge servers, even the central cloud servers when the edge computing resource is insufficient. Moreover, the computing capacity of the D2D-enabled MEC networks is larger than that of the MEC networks since the computing resource of the MDs can be used for processing the offloaded computing tasks. Finally, the upper bounds

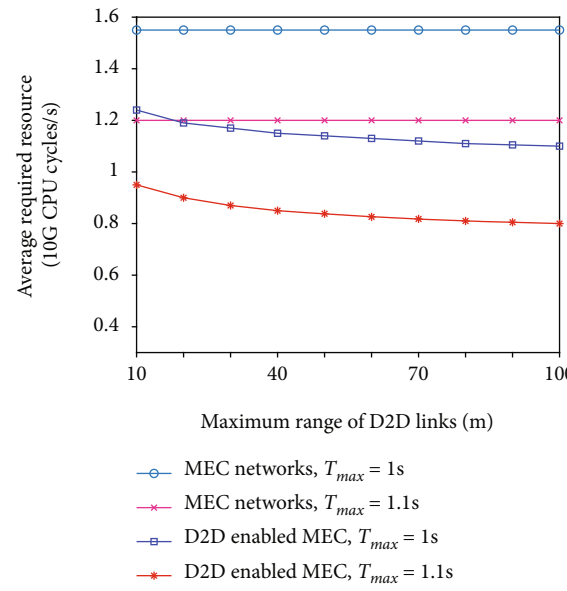

Figure 5: The average required edge computing resource vs. the maximum range of D2D links.

of the computing capacity of the D2D-enabled MEC networks and the MEC networks are the same, that is the overall number of MDs, i.e., 80. The result indicates that employing D2D communications can notably save the required computing resources from edge data center or cloud data center.

Figure 4 depicts the access ratio. It is the proportion of MDs that can be supported versus the total number of MDs. It can be observed that the access rates of both the MEC networks and D2D-enabled MEC networks decreasing with the number of MDs due to the limited edge computing resources, while the access rate of the D2D networks will correspondingly increase. The reason is that, the allocated edge computing resources for each MD in the MEC networks and D2D-enabled MEC networks decreases with the total number of MDs, which results in the smaller proportion of MDs that can be supported. Furthermore, the MDs computing resources can be fully utilized as the total number of MDs in the D2D networks increase, which eventually improves the access rate. We can also observe that the access rate of the D2D-enabled MEC networks decreases slower than that of the MEC networks since it utilizes the MD computing resources for capacity enhancement.

Figure 5 illustrates the average required computing resources for each MD, versus the maximum range of D2D links in the MEC networks and D2D-enabled three-tier MEC networks, respectively. It shows that the average required computing resources in the D2D-enabled MEC networks are smaller than those in the MEC networks, especially when the maximum range of D2D links becomes large. This is because each MD has more potential neighboring MDs to offload tasks as the range of D2D link becomes larger. In this way, the computing resources of all MDs can be further utilized via D2D offloading. We can also observe a reduced required computing resource with the decreased delay tolerance in both networks.

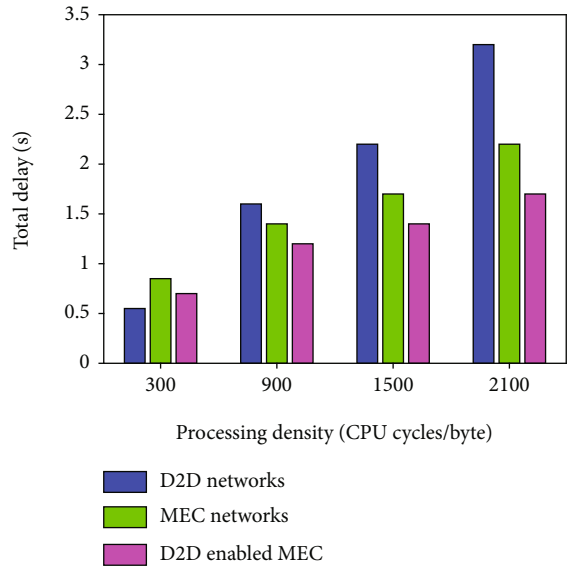

Figure 6: Total delay with different processing densities.

Figure 6 presents the total delay with different computation processing densities. Comparing the proposed scheme with two benchmarks (i.e., MEC networks and D2D networks), we can observe the following: when the processing density is low, the task processed at the device tier performs well compared to offloading to the edge cloud or central cloud DC. This is because the transmission latency on account of offloading from the MD to the edge node or cloud node dominates over the local processing time at the local device tier.

Nevertheless, as the processing density increases, the proposed scheme outperforms the D2D processing at the device tier. Furthermore, with higher processing density, only the MEC networks may not be able to meet the low-latency requirement. On the other hand, as our proposed D2D-enabled MEC networks optimize the offloading decisions, we obtain a significantly lower total latency compared to the other solutions.

7. Conclusion

In this article, we consider cooperative offloading in D2D-enabled three-tier MEC networks as a part of the IoT. By leveraging the advantages of both MEC and D2D cooperation techniques, the computing resources in edge nodes, cloud nodes, and D2D MDs can be effectively utilized. We jointly optimize the task offloading decisions and computing resource allocation to minimize the total delay of tasks. The formulated problem is a mixed-integer nonlinear optimization problem. When networks grow larger, it is difficult to be solved. To work out this problem, we adopt a parallel optimization framework based on the ADMM algorithm. Simulation results show that the proposed framework and solution outperform the other solutions with various parameters. In our future work, we may consider the queue of tasks, partible offloading, and privacy protection when investigating the offloading problem.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Nature Science Foundation of China Projects (No. 61871051).

References

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

[2] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[3] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2017.

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

[5] N. Chen, B. Rong, X. Zhang, and M. Kadoch, "Scalable and flexible massive MIMO precoding for 5G H-CRAN," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 46–52, 2017.

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

[7] K. Zhang, Y. Mao, S. Leng et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[8] M. Li, Q. Wu, J. Zhu, R. Zheng, and M. Zhang, "A computing offloading game for mobile devices and edge cloud servers," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 2179316, 10 pages, 2018.

[9] E. Cuervo, A. Balasubramanian, D.-k. Cho et al., "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10*, pp. 49–62, San Francisco, California, USA, June, 2010.

[10] Ning Zhang, Nan Cheng, A. T. Gamage, Kuan Zhang, J. W. Mark, and Xuemin Shen, "Cloud assisted hetnets toward 5G wireless networks," *IEEE Communications Magazine*, vol. 53, 6-Supplement, pp. 59–65, 2015.

[11] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in *2013 Proceedings IEEE INFOCOM*, Turin, Italy, 2013.

[12] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing towards balanced

delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

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

[14] A. Mukherjee, D. de, and D. G. Roy, "A power and latency aware cloudlet selection strategy for multi-cloudlet environment," *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 141–154, 2019.

[15] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[16] A. Ewaisha and C. Tepedelenlioglu, "Offloading deadline-constrained cellular traffic," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pp. 1447–1451, Pacific Grove, CA, USA, October 2018.

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

[18] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and CPU time allocation for mobile edge computing," in *IEEE GLOBECOM*, pp. 1–6, Washington, DC, USA, December 2016.

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

[20] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[21] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

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

[23] Y. He, J. Ren, G. Yu, and Y. Cai, "Joint computation offloading and resource allocation in D2D enabled MEC networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019.

[24] J. Feng, L. Zhao, D. Jianbo, X. Chu, and F. R. Yu, "Computation offloading and resource allocation in D2D-enabled mobile edge computing," in *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, May 2018.

[25] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 64–71, 2017.

[26] S. Josilo and G. Dan, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 85–97, 2019.

[27] C. Zheng, Z. Zhou, T. Ristaniemi, and Z. Niu, "Energy efficient optimization for computation offloading in fog computing system," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Singapore, December 2017.

[28] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, 2018.

[29] H. Huang, J. Mu, and X. Jing, "Cooperative spectrum sensing based on centralized double threshold in mcn," *China Communications*, vol. 17, no. 5, pp. 235–242, 2020.

[30] C. Wang, C. Liang, F. R. Yuv, Q. Chen, and L. Tang, "Joint computation offloading, resource allocation and content caching in cellular networks with mobile edge computing," in *2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017.

[31] J. Mu, X. Jing, J. Xie, and Y. Zhang, "Multistage spectrum sensing scheme with SNR estimation," *IET Communications*, vol. 13, no. 9, pp. 1148–1154, 2019.

[32] B. Ji, Z. Chen, S. Chen, B. Zhou, C. Li, and H. Wen, "Joint optimization for ambient backscatter communication system with energy harvesting for IoT," *Mechanical Systems and Signal Processing*, vol. 135, pp. 106412–106412.10, 2020.

[33] T. Q. Thinh, J. Tang, Q. D. la, and T. Q. S. Quek, "Offloading in mobile edge computing: task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[34] B. Rong, Y. Qian, and K. Lu, "Integrated downlink resource management for multiservice wimax networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 621–632, 2007.

[35] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[36] Z. Liu, J. Zhang, and W. Jingyan, "Joint optimization of server placement and content caching in mobile edge computing networks," in *Proceedings of the 2019 8th International Conference on Networks, Communication and Computing*, pp. 149–153, Luoyang, China, December 2019.

[37] M. Mukherjee, S. Kumar, M. Shojafar, Q. Zhang, and C. X. Mavromoustakis, "Joint task offloading and resource allocation for delay-sensitive fog networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, Shanghai, China, May 2019.

[38] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: an admm framework," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2763–2776, 2019.

[39] T. Liang, L. Yong, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, San Francisco, CA, USA, April 2016.

[40] J. du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[41] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4193–4207, 2019.

[42] C. Kachris and I. Tomkos, "A survey on optical interconnects for data centers," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1021–1036, 2012.

[43] Z. Liu, J. Zhang, Y. Li, L. Bai, and Y. Ji, "Joint jobs scheduling and lightpath provisioning in fog computing micro datacenter networks," *Journal of Optical Communications and Networking*, vol. 10, no. 7, pp. B152–B163, 2018.

[44] Y. Ji, R. Gu, Z. Yang, J. Li, H. Li, and M. Zhang, "Artificial intelligence-driven autonomous optical networks: 3S architecture and key technologies," *Science China Information Sciences*, vol. 63, no. 6, 2020.

[45] I. Komnios, F. Tsapeli, and S. Gorinsky, "Cost-effective multimode offloading with peer-assisted communications," *Ad Hoc Networks*, vol. 25, pp. 370–382, 2015.

[46] Y. Cui, J. Song, K. Ren et al., "Software defined cooperative offloading for mobile cloudlets," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1746–1760, 2017.

[47] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[48] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2017.

[49] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: an environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, 2018.

[50] S. P. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[51] H. K. Nguyen, Y. Zhang, Z. Chang, and Z. Han, "Parallel and distributed resource allocation with minimum traffic disruption for network virtualization," *IEEE Transactions on Communications*, vol. 65, no. 3, pp. 1162–1175, 2017.

[52] E. Che, H. D. Tuan, and H. H. Nguyen, "Joint optimization of cooperative beamforming and relay assignment in multi-user wireless relay networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 10, pp. 5481–5495, 2014.

[53] R. Horst and N. V. Thoai, "DC programming: overview," *Journal of Optimization Theory and Applications*, vol. 103, no. 1, pp. 1–43, 1999.

[54] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge Univ. Press, Cambridge U. K., 2004.

[55] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[56] M. H. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, pp. 6790–6805, 2018.

[57] Y. Mao, J. Zhang, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, USA, December 2016.

[58] Y. He, J. Ren, G. Yu, and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1750–1763, 2019.