D2D-Enabled Mobile-Edge Computation Offloading for Multiuser IoT Network

Yuhan Yang, Chengnian Long, *Senior Member, IEEE*, Jing Wu, *Senior Member, IEEE*, Shaoliang Peng, *Member, IEEE*, and Bo Li, *Fellow, IEEE*

***Abstract*—The emerging mobile-edge computing paradigm provides opportunities for the resource-hungry mobile devices (MDs) to migrate computation. In order to satisfy the requirements of MDs in terms of latency and energy consumption, recent researches proposed diverse computation offloading schemes. However, they either fail to consider the potential computing resources at the edge, or ignore the selfish behavior of users and the dynamic resource adaptability. To this end, we study the computation offloading problem and take into consideration the dynamic available resource of idle devices and the selfish behavior of users. Furthermore, we propose a game theoretic offloading method by regarding the computation offloading process as a resource contention game, which minimizes the individual task execution cost and the system overhead. Utilizing the potential game, we prove the existence of Nash equilibrium (NE), and give a lightweight algorithm to help the game reach a NE, wherein each user can find an optimal offloading strategy based on three contention principles. Additionally, we conduct analysis of computational complexity and the Price of Anarchy (PoA), and deploy three baseline methods to compare with our proposed scheme. Numerical results illustrate that our scheme can provide high-quality services to users, and also demonstrate the effectiveness, scalability and dynamic resource adaptability of our proposed algorithm in a multiuser network.**

***Index Terms*—Computation offloading, device-to-device (D2D) link, game theory, Nash equilibrium (NE).**

I. Introduction

With the rapid development of the Internet of Things (IoT) and 5G technology [1], there is an increasing need for computation-intensity and time-sensitive tasks, e.g., face recognition, natural language processing and document clustering [2], [3], which often requires low energy consumption, low latency and high reliability. The traditional cloud computing cannot process the huge amounts of data collected from the edge nor provide real-time computation, hence certain tasks need to be processed at the edge devices. However, given the energy and computing capacity constraints, there is still a significant challenge fulfilling the high demand application requirement with the resource-limited devices.

To overcome this challenge, mobile-edge computing (MEC) [4], [5] emerges as a promising solution in recent years. As a complement of cloud computing, MEC proposes to engage computing, storage and control at the network edge. Considering that the mobile-edge servers are close to the data source, the computing task can be performed at the edge rather than be transmitted to the cloud server, which reduces the transmission pressure of the core network, and substantially mitigates the computational burden in the cloud. Unlike the conventional content delivery network (CDN) focusing on accelerating content delivery, the essential issue in MEC is computation migration. This enables mobile devices (MDs) to offload tasks to servers at the edge, esp. computation-intensive or/and delay-sensitive tasks. Hence, to allocate communication and computation resources among a potentially large number of MDs for task offloading, becomes a critical issue. This is complicated by the availability of abundant computing resources at the MDs themselves, which can also be the targets for task offloading by leveraging the device-to-device (D2D) technology [6]. This further helps to enhance the computing capacity, expand the communication coverage and reduce transmission latency. It is natural to consider both MEC and D2D offloading in the resource allocation problem.

Nevertheless, the integration of the two technologies faces challenges described as follow.

1) *QoS-Aware Computation Service in Heterogeneous Network:* Users offload the computation-intensity tasks to MEC servers or nearby idle devices for high Quality of Service (QoS), however, it is a challenge to provide the QoS-aware computation service, and minimize the overhead of the system in the meanwhile. This is due to the heterogeneity in edge server computing capability, network condition and user demands. Existing work [7] satisfied high QoS for users, relies on a prior agreement of the maximum available resource between users and the idle devices (referred as mobile cloudlets), which is difficult to deploy in a multiuser heterogeneous network. Therefore, the computation offloading scheme should

Manuscript received August 10, 2020; revised January 6, 2021; accepted March 20, 2021. Date of publication March 25, 2021; date of current version August 6, 2021. This work was supported in part by NSFC under Grant 62073215, Grant 61873166, and Grant 61673275; in part by the Science and Technology Commission, Shanghai Municipality under Grant 19511102102; in part by the Shanghai Automotive Industry Technology Development Foundation under Grant 2008; in part by Research Grants Council (RGC) Research Impact Fund (RIF) under Grant R6021-20; and in part by RGC General Research Fund (GRF) Grants under Contract 16207818 and Contract 16209120. *(Corresponding authors: Jing Wu; Chengnian Long.)*

Yuhan Yang, Chengnian Long, and Jing Wu are with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China, and also with the Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai 200240, China (e-mail: yuhanyang@sjtu.edu.cn; longcn@sjtu.edu.cn; jingwu@sjtu.edu.cn).

Shaoliang Peng is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: slpeng@hnu.edu.cn).

Bo Li is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: bli@cse.ust.hk).

Digital Object Identifier 10.1109/JIOT.2021.3068722

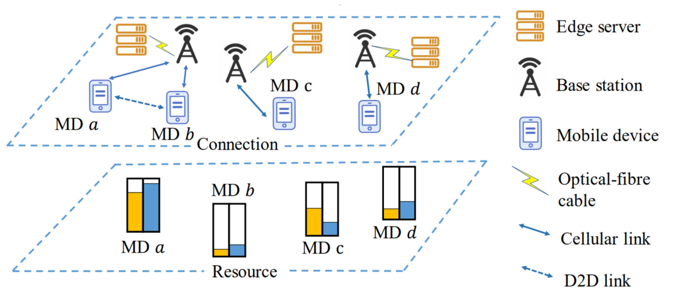


Fig. 1. D2D-enabled mobile-edge computation offloading framework. Cellular link and D2D link are illustrated in the connection layer, and the resource layer shows the available computation and communication resource in each device, where the yellow (blue) portion represents the available communication (computation) resource.

make a balance between system overhead and individual cost, and only depend on the current system information.

2) *Dynamic Resource Adaptability in Unpredictable Environment:* In a D2D-enabled computation offloading system, the availability of the resource in idle devices is variable and uncontrolled, which causes the available computation resource is not static. Moreover, the D2D offloading strongly relies the cooperation among users, and the individual selfish behavior may influence the offloading strategy [8]. Hence, how to maintain and promote the cooperative offloading is a critical problem, which also indicates that the resource allocation scheme should be scalable to multiple users.

The above challenges motivate us to propose a distributed lightweight computation offloading scheme for terminal users. In this article, we introduce a D2D-enabled mobile-edge computation offloading framework, and study the MEC offloading and D2D offloading in a multiuser environment. As illustrated in Fig. 1, an MD can offload its task to the edge server through the base station (BS), or an idle MD with sufficient computing resources. The main contributions are summarized as follows.

1) Considering the availability of idle devices is variable and uncontrolled, meanwhile, the D2D offloading may fail due to communication limitation or the selfish behavior of users. We study and formulate the cooperative computation offloading problem by considering the QoS demand in terms of processing time and energy consumption. Moreover, a cooperation constraint based on "mirror incentive" is proposed to promote cooperation among users.
2) We propose a game theoretic method to address the computation offloading problem by modeling the offloading process as a resource contention game, where three contention principles are introduced to facilitate the contention process. To maintain the strategy update and computation offloading, an interaction flow between users and the BSs is provided. We further prove the existence of Nash equilibrium (NE) and conduct the analysis of the computational complexity and the PoA, which demonstrates that the impact of users' selfish behavior on system overhead is marginal.
3) For performance evaluation, we give an enumeration-based algorithm as a baseline algorithm, which can find a globally optimal offloading strategy combination with prior knowledge. Moreover, we deploy two offloading methods for performance comparison. Simulation results demonstrate the effectiveness, convergence behavior, scalability and adaptability of our algorithm.

The remainder of this article is organized as follows. The related work is discussed in Section II. We introduce the system model and the computation offloading scheme in Sections III and IV. In Section V, we demonstrate the performance evaluation and discuss about our proposed scheme. Finally, the conclusion is given in Section VI.

II. Related Work

Computation offloading in MEC has been extensively studied, which can be roughly divided into MEC offloading and D2D-enabled hybrid offloading. For the first category, the existing works pay attention to investigate the MEC architecture [9], [10], efficient data routing for IoT communication [11], [12], and further study the resource allocation problem in a MEC system [13]–[19]. Guo *et al.* [13] provided a dynamic resource scheduling policy to reduce the task completion time and energy consumption based on mobile cloud computing architecture, wherein the task in one device is divided into various subtasks and offloaded to the servers. Lyu *et al.* [14] considered the joint optimization of communication and computation resources, and presented a heuristic offloading decision algorithm by decomposing the resource allocation problem into two subproblems. Other schemes are proposed in [15]–[17], which focuses on the single-user scenario and does not consider the balance between individual overhead and system overhead. To scale to multiple users, Chen *et al.* [18] introduced a comprehensive framework and designed a trustful scheme for resource allocation. Tran and Pompili [19] proposed to use convex and quasi-convex optimization techniques to address the resource allocation problem, and use a heuristic algorithm to solve the task offloading problem in a multicell MEC system. These works extended the limited capacity of terminal devices in terms of storage, computation and battery lifetime by computation migration, however, a mass of potential unused resource in the nearby devices is ignored.

To exploit the unused resource in the edge, a number of works focus on the D2D communication-assisted offloading [20]–[25]. Wen *et al.* [20] attempted to improve the energy-efficient communication by introducing D2D technology, Yang *et al.* [21] investigated the incentive computation offloading problem in fog-enabled D2D network, and the hybrid offloading in terms of MEC offloading and D2D offloading is considered in [22] and [23]. Moreover, to enhance the system efficiency, Pu *et al.* [24] addressed resource sharing between MDs based on Lyapunov optimization method in fog computing architecture. Yu *et al.* [25] designed a D2D multicast computation offloading framework for minimizing the overall energy consumption. From the perspective of individual behavior and cost, a number of works addressed the resource allocation problem based on game theory [26]–[30]. Liu *et al.* [26] discussed the task migration model for vehicular edge network. Ma *et al.* [27] proposed a computation offloading algorithm based on cloudlet, and considered the scanning and maintaining energy in the system model. Shah-Mansouri and Wong [28] designed a dynamic resource allocation scheme to reassign the processing power in MEC servers. Furthermore, Zheng *et al.* [29] introduced a stochastic game approach for mobile users that with random computation tasks. The above researches developed various distributed method to promote the D2D offloading, whereas the in-depth discussion about the cooperation among users and the corresponding impact on the system are ignored.

Generally speaking, existing MEC offloading works focus on the optimization of the joint communication and computation resource allocation problem, which enhances the quality of computation service for multiple users, but a potentially large number of resource in the idle devices is unused. Fortunately, this problem is addressed in hybrid offloading through combining D2D technology, which leverages the resource in idle devices opportunistically. However, the existing works are not adaptable to an unknown heterogeneous network, wherein the computation resource and the demands of users are not always predictable. Moreover, they also fail to discuss the influence of cooperative offloading strategy and system overhead caused by user mobility and individual behavior. Therefore, we propose a D2D-enabled MEC offloading scheme in this article, which considers the dynamic resource in devices and the individual willingness in cooperation. Furthermore, we develop a game theoretic offloading algorithm that only relies on the current information sharing, and evaluate the individual offloading cost and system overhead under three different resource contention principles.

III. System Model

A. D2D-Enabled MEC Framework

The computation offloading is implemented in a multiuser dense IoT network, which introduces a dense deployment of small cells in the edge to get the access nodes as close as possible to the end users [31]. In this section, we introduce the D2D-enabled MEC offloading framework as shown in Fig. 1. We consider a multicell dense network that involves a set $\mathcal{M} = \{1, 2, \ldots, M\}$ of MDs and a set $\mathcal{S} = \{1, 2, \ldots, S\}$ of BSs. The MEC servers are deployed in the proximity of the BSs via high-speed optical-fibre cable. In addition, let $\mathcal{N} = \{1, 2, \ldots, N\}(\mathcal{N} \subset \mathcal{M})$ denotes the set of idle devices that are called mobile helpers (MHs) and provide computation support to other MDs. Both of the MEC servers and idle devices have ability to provide computing service to the MDs that have computation tasks, however, the number of $\mathcal{N}$ is dynamically changing. Each MEC server can serve more than one MD, while only one D2D link can be established between two MDs due to the limited computation capacity in the idle device.

B. Task Model

We use $T_i = (D_i, O_i, C_i, L_i)$ to denote the task in MD i, where D_i and O_i are the input and output data size, C_i is the required CPU cycles that is determind by the processing density γ (i.e., $C_i = \gamma \cdot D_i$), and L_i is the acceptable latency.

We use $\Lambda = \{-M, \ldots, -1, 0, 1, \ldots, S\}$ to denote the offloading object, and $\pi_{i,j} \in \{0, 1\}$ to denote the offloading decision of MD i, where $i \in \mathcal{M}$, and $j \in \Lambda$. To simplify expression, let $\bar{\mathcal{M}} = \{-M, \ldots, -1\}$. If MD i executes the task locally, $j = 0$ and $\pi_{i,j} = 1$. We assume that $\pi_{i,-i} = 0$, which is in conflict with $\pi_{i,0}$. For $j \in \bar{\mathcal{M}}$, $\pi_{i,j} = 1$ indicates that MD i decides to offload task to an idle MH j. For $j \in \mathcal{S}$, $\pi_{i,j} = 1$ indicates that MD i chooses offloading to the MEC server connected to BS j. More notation definitions are shown in Table I. In the rest of this article, we directly use server j to represent the server connected to BS j. Moreover, We use $\mathcal{A}_i = \{(\pi_{i,-M}, \ldots, \pi_{i,0}, \ldots, \pi_{i,S}) | \pi_{i,j} \in \{0, 1\}\}$ to denote the offloading strategy of MD i, and $\mathcal{A}_{-i}$ to denote the offloading strategies of other MDs except MD i. For an idle device that does not have task to perform, we assume that the strategy of it is local computing. Since MD i can only connect to a BS via cellular link or connect to an MH via D2D link, we have $\sum_{j \in \Lambda} \pi_{i,j} = 1$ to ensure that each task is offloaded to at most one server or one idle device.

C. Cost Model

In this section, we introduce the cost model for each MD in three computation scenarios: 1) *local computing; 2) MEC offloading; and 3) D2D offloading*. To provide high QoS of each MD, we build the cost model in terms of delay and energy consumption.

1) Local Computing: If MD i executes the task T_i locally, i.e., $\pi_{i,j} = 1$ and $j = 0$, the total cost includes execution time and execution energy. We denote by f_i^M the local computing capacity of MD i, and the execution time is $T_{i,j}^{\text{exe}} = (C_i / f_i^M)$. We denote by ε the energy coefficient to represent the energy consumption per CPU cycle. The execution energy can be calculated as $E_{i,j}^{\text{exe}} = \varepsilon \cdot C_i$, and we obtain the total cost of local computing as

$$Q_{i,j}^L = \alpha_i \cdot T_{i,j}^{\text{exe}} + \beta_i \cdot E_{i,j}^{\text{exe}} \tag{1}$$

where α_i and β_i represent the coefficients of execution time and execution energy, which can be set according to users' interests. If the task is latency-aware and the device has continuous energy supply, the weight of task execution delay α_i can be close to 1. It means that the user needs a computation service with low latency. If the device is resource-hungry and the task is energy-intensive, the energy consumption will account for a larger proportion of the total cost and the value of β_i will be higher than α_i. Since the type of task depends on the capacity of device and the demand of user, we consider that the values of the two parameters are time-varying but uncontrollable, which can be set by users.[1]

2) MEC Offloading: MD i can establish cellular link with the BS and offload task to the MEC server j, i.e., $\pi_{i,j} = 1$

[1]A certain normalization method should be adopted due to the different magnitude and unit of execution delay and energy consumption.

TABLE I
NOTATION DEFINITION

Notation	Definition
$\mathcal{M}$	Set of mobile devices
$\mathcal{B}$	Set of base stations
$\mathcal{N}$	Set of idle devices
T_i	Task in MD i
D_i	Input data size for T_i
O_i	Output data size for T_i
C_i	Required CPU cycles for T_i
L_i	Acceptable latency for T_i
$\pi_{i,j}$	Offloading indicator of MD i
γ	Task processing density
ε	Energy coefficient of MD i
α_i	Delay coefficient of MD i
β_i	Energy consumption coefficient of MD i
Λ	Set of offloading objects
$\mathcal{A}_i$	Offloading strategy of MD i
f_i^M	Computing capacity of MD i
f_j^S	Computing capacity of server j
P_i	Transmission power of MD i
W	Sub-channel bandwidth
$r_{i,j}^{BS}$	Uplink data rate of MD i
$r_{i,j}^{D}$	Data rate between MD i and MD j
H_i	Channel gain in cellular link
$H_{i,j}$	Channel gain between MD i and MD j
N_0	Noise power
$Con_j(k)$	Contribution of MD j to MD i in the k-th Cooperation
Con_j	Long-run cooperation credit of MD j
$\overline{Con}$	Contribution threshold for initiation
$E_j^{con}(k)$	Energy consumption of MD j in the k-th cooperation
E_j^{max}	Energy consumption limitation of MD j

and $j \in \mathcal{S}$. In this case, the transmission time in wireless link and the computation time in MEC server, are commonly used to evaluate the major delay. Moreover, task transmitting process will consume the energy of MD, which is determined by the transmission power. Hence, the offloading cost will be modeled from the three perspectives above. First, we focus on the transmission process, where an orthogonal frequency-division multiple access (OFDMA) method is adopted for channel resource allocation. In this article, we consider to allocate one subchannel for each MD that offloads task via cellular link or D2D link. Then the uplink data rate of MD i defined by $r_{i,j}^{BS}$ can be shown as follows $r_{i,j}^{BS} = W\log_2(1+[P_iH_i/N_0])$, where W is the subchannel bandwidth in cellular link. N_0 and P_i denote the noise power and the transmission power of MD i, respectively. H_i denotes the channel gain in cellular link. Then the transmission time can be calculated by $T_{i,j}^{\text{trans}} = D_i/r_{i,j}^{BS}$. Considering that the downlink date rate is high enough because of the high transmission power of BS, we omit the transmission time in the downlink. Based on $T_{i,j}^{\text{trans}}$, we further denote by P_i the transmission power of MD i, and the transmission energy consumption can be given as $E_{i,j}^{\text{trans}} = P_i \cdot T_{i,j}^{\text{trans}}$.

In the task performing process, since the server may serve more than one MD simultaneously, the waiting time in the execution queue should be considered. We develop the M/M/1 queue model to compute the queuing time in each independent server, where the arrival of the task in the MEC server is a Poisson process, and the distribution of service time follows exponential distribution. Following [32], we can derive the average arriving rate in server j as $\lambda_j = \sum_{i\in\mathcal{M}} \pi_{i,j} \cdot C_i$. We denote the available computation capacity of server j as f_j^S, the computation delay including the waiting time and the execution time is as follows:

$$T_{i,j}^{\text{exe}}(\mathcal{A}_i, \mathcal{A}_{-i}) = \frac{C_i}{f_j^S - \lambda_j}. \tag{2}$$

Note that the average arriving rate is based on the required CPU cycles of tasks that are offloaded to server j, the BS can collect these necessary information and report the arriving rate to each MD. The total cost in MEC offloading is

$$Q_{i,j}^M(\mathcal{A}_i, \mathcal{A}_{-i}) = \alpha_i \cdot \left(T_{i,j}^{\text{trans}} + T_{i,j}^{\text{exe}}\right) + \beta_i \cdot E_{i,j}^{\text{trans}}. \tag{3}$$

3) D2D Offloading: In D2D offloading, one D2D link is established between an idle MH j and a busy MD i. We regard the BS as a manager to be responsible for the D2D offloading and allocate subchannel for D2D pairs. Moreover, both of the MDs in D2D pair are required to connect to the same BS. Similar to the cellular link, we can calculate the data rate in D2D link as $r_{i,j}^D = W\log_2(1+[P_iH_{i,j}/N_0])$, where $H_{i,j}$ is the channel gain between MD i and MD j. The data rate is not equivalent in D2D communication and the transmission power of terminal device is lower than that of BS, hence the output delay in transmission time is considered, which is shown as follows:

$$T_{i,j}^{\text{trans}} = \frac{D_i}{r_{i,j}^D} + \frac{O_i}{r_{j,i}^D} \tag{4}$$

where $r_{i,j}^D \neq r_{j,i}^D$, and the transmission energy consumption can be given as $E_{i,j}^{\text{trans}} = P_i \cdot (D_i/r_{i,j}^D)$.

Given that only one D2D link can be established between two MDs, the execution time in MH j is $T_{i,j}^{\text{exe}} = (C_i/f_j^M)$. Then the total cost in D2D offloading is given as

$$Q_{i,j}^D = \alpha_i \cdot \left(T_{i,j}^{\text{trans}} + T_{i,j}^{\text{exe}}\right) + \beta_i \cdot E_{i,j}^{\text{trans}}. \tag{5}$$

Based on the cost model in terms of local computing, MEC offloading and D2D offloading, the cost function of MD i under different strategies can be obtained as

$$Q_i(\mathcal{A}_i, \mathcal{A}_{-i}) = \begin{cases} Q_{i,j}^L \cdot \pi_{i,j}, & \pi_{i,j} = 1, j = 0 \\ \sum_{j\in\mathcal{S}} Q_{i,j}^M(\mathcal{A}_i, \mathcal{A}_{-i}) \cdot \pi_{i,j}, & \pi_{i,j} = 1, j \in \mathcal{S} \\ \sum_{\{j\in\bar{\mathcal{M}}\}\setminus\{j=i\}} Q_{i,j}^D \cdot \pi_{i,j}, & \pi_{i,j} = 1, j \in \bar{\mathcal{M}}. \end{cases} \tag{6}$$

D. Cooperation Constraint for D2D Offloading

In practical scenario, MDs are independent and selfish devices, that is, they may not have enough motivation to share resource. For an idle device, it is impossible to contribute resource to others in vain, unless there is additional benefit. Therefore, to promote the cooperative among MDs, an cooperation constraint for D2D offloading based on the resource contribution is proposed in this section.

Recalling the D2D offloading cost model, the major energy consumption is caused by computation and transmission. We propose the constraint based on "mirror incentives," which means that the computation resource obtained by each MD through D2D offloading depends on what it has contributed to others. Specifically, we introduce a long-run contribution metric Con_j for an MD j as the cooperation credit. If MD j provides computation support for MD i, based on the law of diminishing marginal utility, the contribution of MD j in a D2D offloading can be defined utilizing the a logarithmic function [33] as follows:

$$\mathrm{Con}_j(k) = f_j^M \log_2(1 + C_i) \tag{7}$$

where k means the kth D2D offloading cooperation of MD j. Then the cooperation credit Con_j in the last cooperation (referred as the Kth cooperation) can be updated by

$$\mathrm{Con}_j = \frac{1}{K} \cdot \sum_{k=1}^{K} \mathrm{Con}_j(k). \tag{8}$$

Moreover, if MD j has task to offload and wants to establish D2D offloading with other MD j' in the next offloading, the credit Con_j should satisfy:

$$\mathrm{Con}_j + \overline{\mathrm{Con}} > \frac{1}{K+1} \cdot \sum_{k=1}^{K+1} \mathrm{Con}_{j'}(k) \tag{9}$$

where j' indicates the idle MD that provides computation support to MD j in the past cooperations and $\mathrm{Con}_{j'}(k) = f_{j'}^M \log_2(1 + C_j)$ is the contribution received by MD j. $\overline{\mathrm{Con}}$ is an initial contribution threshold designed for the first round of D2D offloading, which permits MDs obtain a small amount of resource freely. The cooperation credit is used to prevent the selfish behavior of free-riding, while motivating idle devices to share the unused resource. Additionally, a maximum of energy consumption in the idle device should be considered to guarantee the requirement of QoS. This is because, the MH does not want to drain off the battery for executing tasks for others, meanwhile, returning a complete computation result should be guaranteed for each MD that has task to offload. Therefore, another cooperation constraint for D2D offloading is

$$E_j^{\mathrm{con}}(k) < E_j^{\max} \tag{10}$$

where $E_j^{\mathrm{con}}(k) = \varepsilon \cdot C_i + P_j \cdot (O_i / r_{j,i}^D)$ is the energy consumption of MD j in the kth cooperation, and $E_j^{\max}$ is the energy consumption limitation for MD j, which ensures the complete execution of task when MD j provides computation service for MD i. We should emphasize that a constant energy supply for an idle device cannot be guaranteed, hence the previous energy consumption should be considered in the constraint. For an idle device (e.g., a PC) that is with limited battery and cannot be recharged in time, the summation of the energy consumption over k cooperation should be smaller than the energy consumption limitation, and (10) is rewritten as $\sum_{\tau=1}^{k} E_j^{\mathrm{con}}(\tau) < E_j^{\max}$. Therefore, we set the energy consumption limitation as a variable, which is regarded as an energy consumption budget and determined by the current energy supply condition. That is, the maximum energy consumption of an MH will decrease after several cooperations if it cannot be recharged in time.

For all D2D pairs, each MD records the offloading activity and the resource received or contributed to others in the local database, then upload it with the current energy consumption limitation $E_j^{\max}$ to BS. The BS, which connects with the two MDs in a D2D pair, is responsible for the D2D pair, hence it will synchronize the offloading records that are uploaded by MDs.[2] Before establishing the D2D link, the BS verifies the cooperation constraints, wherein legitimate D2D pair can be established and maintained for offloading.

E. Problem Formulation

Based on the analysis of task model, cost model and cooperation constraint, we formulate the computation offloading problem in the D2D-enabled MEC system. In order to provide high QoS to resource-limited users and assist to perform the time-sensitive tasks, we focus to minimize the overall offloading cost in terms of processing delay and energy consumption. The mathematical optimization problem can be formulated as follows:

$$\begin{aligned} \min \quad & \sum_{i\in\mathcal{M}} Q_i(\mathcal{A}_i, \mathcal{A}_{-i}) \\ \text{s.t.} \quad & C1: \sum_{j\in\Lambda} \pi_{i,j} = 1 \quad \forall i \in \mathcal{M} \\ & C2: \sum_{i\in\mathcal{M}} \frac{C_i}{L_i} \cdot \pi_{i,j} \le f_j^S \quad \forall j \in \mathcal{S} \\ & C3: \mathrm{Con}_i + \overline{\mathrm{Con}_i} > \mathrm{Con}_{i'}(k) \quad \forall i, i' \in \mathcal{M} \\ & C4: E_i^{\mathrm{con}} < E_i^{\max} \quad \forall i \in \mathcal{M}. \end{aligned} \tag{11}$$

Here, constraint $C1$ requires that each MD can only choose one offloading strategy. Specially, there is at most one D2D link established between two MDs when MD i chooses D2D offloading. Moreover, constraint $C2$ ensures the total required computation resource of server j is less than its capacity. $C3$ and $C4$ are the cooperation constraints in D2D offloading.

IV. Computation Offloading Scheme for Multiuser

The main object of (11) is to minimize the offloading cost of all MDs that have tasks to perform, wherein the constraints of resource, communication and cooperation are considered. To solve it, we introduce a method that relies on the prior knowledge about the system, and propose a game theoretic method based on the online cooperation among users.

A. Enumeration-Based Offloading Method With Prior Knowledge

In this section, we introduce an enumeration-based offloading algorithm (EBOA) based on prior knowledge of system

[2]In this article, we assume that the MDs record the offloading process honestly, and the records uploaded to the BS are credible. The credits maintained in the BS depends on the contribution records submitted by the two MDs. If there is difference between the two sets of information, the BS should calculate the final credit according to a certain statistical method.

state and resource availability. In EBOA, a controller is necessary (e.g., the BS) to collect the tasks of each MDs and the availability of idle devices. According to the number of available MHs and the cooperation credit of each MD, the controller calculates the reachable offloading combinations, then find an overall offloading decisions with minimum cost for each MD. Specifically, the controller executes the procedure as follows.

1) Collects the task of each MD and the resource availability of idle devices.
2) Calculates the reachable offloading strategies based on the constraints of communication and cooperation.
3) Enumerates all offloading decisions based on the reachable offloading combinations and choose an optimal overall offloading strategy that satisfies (11).
4) Broadcasts the optimal strategy for each MD.

The algorithm EBOA can help to find a globally optimal offloading strategy combination by traversing the whole strategy space, however, the computational complexity is considerably high. After collecting all task parameters and the resource availability of idle devices, the controller executes the second step to obtain all reachable offloading strategy combinations that satisfy the constraints in (11). Recall that each MD has at most N D2D offloading strategies and S MEC offloading strategies, hence there is $N+S+1$ strategies (including local computing) for each MD, and the running time of the second step is $O((N+S)^{M-N})$. In the third step, the controller needs to traverse the whole offloading strategy space to find a globally optimal strategy combination with the minimum total cost, the computational complexity is also exponential (i.e., $O((N+S)^{M-N})$). By adding the complexity of the two steps, we obtain the computational complexity of EBOA is $O((N+S)^{M-N})$. Obviously, the algorithm EBOA is not flexible and scalable as the number of MDs increases. Moreover, it relies on prior knowledge about the system and a centralized controller, which makes it difficult to deploy in a complex and dynamic environment. That is, the reachable strategies combinations will change according to the resource availability of idle devices and the cooperation history among users. Therefore, the algorithm EBOA may be unworkable in a dynamic IoT network, although it can help to find the optimal offloading strategy combination with the minimum total cost through enumerating all offloading strategies, which can be a baseline to verify the effectiveness of our proposed scheme.

B. Game Theoretic Offloading Method

In this section, we propose a distributed offloading scheme base on game theory for a dynamic network. Considering that the task queue sizes of each MD and server may be unclear, and an equal allocation method does not make full use of resource. In our proposed scheme, MDs with computation tasks will contend the resource in the servers or idle devices, which can be regarded as a resource contention game. Specifically, for obtaining an optimal strategy, each MD collects strategies of other MDs and asks for the resource availability of idle devices from the BS. After finding an optimal strategy, it will send the optimal strategy to the BS to contend resource. The BS, as a manager, determines which MD wins the chance for strategy update and sends a signal to confirm with the winner. Additionally, a slotted time structure is considered for strategy update.

1) Game Definition and Analysis: In the resource contention game among multiple MDs, each MD observes the same information (e.g., their offloading strategies, the availability of idle MHs) and exchange information with the BS. In such a cooperative game, MDs share the offloading strategies but contend for the computation resource. Let $\Gamma = (\mathcal{M}, \{\mathcal{A}_i\}_{i\in\mathcal{M}}, \{Q_i\}_{i\in\mathcal{M}})$ represent the game, where $\mathcal{M}$ is the set of players, $\mathcal{A}_i$ is the strategy set for MD i as defined in *Task Model* and $Q_i(\mathcal{A}_i, \mathcal{A}_{-i})$ is the cost function.

All of the MDs aim to decrease the cost function and obtain high QoS, meanwhile, they are selfish and rational to choose offloading object. Each MD collects enough information from the BS and other MDs, and calculate the unique strategy once the cost function is determined. Therefore, the game is a pure strategy game, where the players only focus on their own profits. Our goal is to design a finite algorithm to help the system reach a stable equilibrium status, where each selfish player obtain a individual optimal strategy. Next, we introduce an important definition of NE in game theory.

Definition 1: A strategy profile $\{\mathcal{A}_i^*\}_{i\in\mathcal{M}}$ is an NE of the game Γ if it satisfies

$$Q_i\left(\mathcal{A}_i^*, \mathcal{A}_{-i}^*\right) \le Q_i\left(\mathcal{A}_i, \mathcal{A}_{-i}^*\right) \quad \forall i \in \mathcal{M} \quad \forall \mathcal{A}_i \in \mathcal{A}. \tag{12}$$

For MD i, if the current strategy profile $\mathcal{A}_i^*$ brings a lower cost and MD i has no incentive to change the strategy. The game Γ reaches to an NE if all of the players have no incentive to change their offloading strategies. Now we show the existence of NE by introducing potential game.

Definition 2: Given a vector $\mathbf{w} = (w_i)_{i\in\mathcal{M}}$ and a function P, a game Γ is called a weight potential game if for $\forall i \in \mathcal{M}$ and $\forall \mathcal{A}_i, \mathcal{A}_i^{'} \in \mathcal{A}$

$$\begin{aligned} Q_i\left(\mathcal{A}_i^{'}, \mathcal{A}_{-i}\right) &- Q_i(\mathcal{A}_i, \mathcal{A}_{-i}) \\ &= w_i\left(P\left(\mathcal{A}_i^{'}, \mathcal{A}_{-i}\right) - P(\mathcal{A}_i, \mathcal{A}_{-i})\right). \end{aligned} \tag{13}$$

Here, $\mathbf{w}$ is called weight determined by the different parameter settings of MDs, and $P(\mathcal{A}_i, \mathcal{A}_{-i})$ is a global potential function, which reflects the global change when one player updates strategy.

Theorem 1: The game Γ is a weighted potential game with a potential function P as follows:

$$\begin{aligned} P(\mathcal{A}_i, \mathcal{A}_{-i}) = &\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{S}} \frac{\pi_{i,j}}{f_j^S} + \sum_{i\in\mathcal{M}}\left(\sum_{j\in\mathcal{M}} \frac{\pi_{i,-j}}{f_j^M} + \frac{\pi_{i,0}}{f_i^M}\right) \\ &+ \sum_{i\in\mathcal{M}} \mathbf{L}_i \cdot \pi_{i,0} + \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{S}} \mathbf{G}_{i,j}(\mathcal{A}_i, \mathcal{A}_{-i}) \cdot \pi_{i,j} \\ &+ \sum_{i\in\mathcal{M}}\sum_{j\in\bar{\mathcal{M}}} \mathbf{R}_{i,j} \cdot \pi_{i,j} \end{aligned} \tag{14}$$

where $\mathbf{L}_i = (\beta_i/\alpha_i) \cdot \varepsilon$, $\mathbf{G}_{i,j}(\mathcal{A}_i, \mathcal{A}_{-i}) = (D_i/[C_i \cdot r_{i,j}^M]) + ([\sum_{i\in\mathcal{M}} \pi_{i,j}]/[\mu_j \cdot c_i]) + (\beta_i/\alpha_i) \cdot ([P_i \cdot D_i]/[C_i \cdot r_{i,j}^M])$

and $\mathbf{R}_{i,j} = (D_i/[C_i \cdot r^D_{i,j}]) + (Q_i/[C_i \cdot r^D_{j,i}]) + (\beta_i/\alpha_i) \cdot ([P_i \cdot D_i]/[C_i \cdot r^D_{i,j}])$

Proof: See the Appendix. ■

We show the game Γ is a $\mathbf{w}$-potential game with $w_i = \alpha_i \cdot C_i$ by proving that (13) is satisfied when a user exchange strategy in the four cases: 1) local computing $\rightarrow$ MEC offloading; 2) local computing $\rightarrow$ D2D offloading; 3) MEC offloading $\rightarrow$ D2D offloading; and 4) exchange D2D offloading object. Furthermore, we give another important property of game Γ as Lemma 1 based on [34].

Lemma 1: The game Γ has the finite improvement property (FIP).

Proof: According to the strategy set $\mathcal{A}$, we denote an improvement path $\gamma = (\gamma_0, \gamma_1, \gamma_2, \ldots,)$ for game Γ, and we have

$$P(\gamma_0) < P(\gamma_1) < P(\gamma_2) < \cdots \qquad (15)$$

since the strategy profile $\mathcal{A}$ is a finite set, γ is finite. ■

Obviously that the game Γ is a finite game due to the finite strategy set $\mathcal{A}$. With Lemma 1, we can obtain that there is a pure-strategy NE in game Γ. Moreover, Lemma 1 indicates that the finite improvement path will end with a maximum of P, and the MDs have no incentive to change strategies at this moment. In other words, the sequence γ will converge to an NE state that is independent of the initial state by unilateral optimization. This property motivates us to design an algorithm to improve the profit of each MD, which can reach an NE after finite iterations.

2) Distributed MEC-D2D Offloading Algorithm: In this section, we propose a distributed MEC-D2D offloading algorithm (DM-DOA) to help each MD find an improvement path and facilitate the resource contention game to reach an NE. Unlike EBOA, the MD that has computation task calculates the optimal strategy by exchanging information with other MDs and the BS, which does not traverses all offloading decisions. Moreover, we assume that the message can be received successfully in the information sharing process, and the interaction flow between BS and MDs is shown in Fig. 2. To administer the D2D pairs and the contention process, the BS is considered as a trustful manager to perform idle device discovery and integrate the necessary information (e.g., the available idle devices and the connection status of D2D pairs) for assisting strategy update. Each idle device will report the available resource to the BS according to its own willingness, and only the idle device that has contributed resources will obtain the computation support from other MDs in the future. For the contention principle, three principles are adopted in the BS.

1) *FIFO Principle:* The BS receives all strategy update requests from MDs, and gives update permission to the first MD in each round of the contention.
2) *Greedy Principle:* The BS receives all strategy update requests from MDs, and greedily selects the winner according to the offloading cost.
3) *Minimizing Potential Function (MPF) Principle:* The BS receives all strategy update requests from MDs, and find the winner with an improvement step that can minimize the potential function. That is, the winner in each round will make the overall cost to decrease with a faster speed.

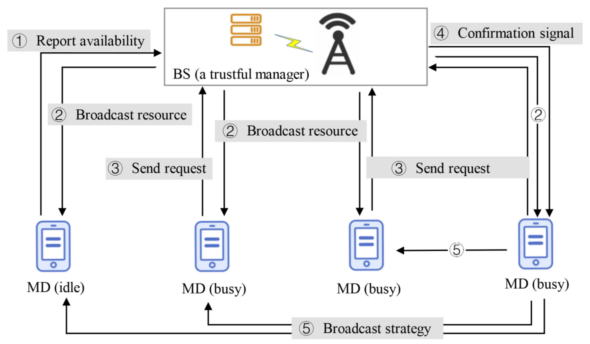


Fig. 2. Message forwarding flow for cooperative offloading.

Algorithm 1: DM-DOA

```
Input: The strategy set $\{\mathcal{A}\}_{i\in\mathcal{M}}$; Parameters for
       calculating the cost of diverse offloading
       strategy($e.g.$, $\alpha_i$, $\beta_i$).
1  for each time slot $t$ do
2      STAGE 1-2: the BS sends the availability of idle
        devices and the current task queue in server;
3      for each MD $i$ do
4          collect messages from the BS and other MDs;
5          if there are available MHs then
6              calculate $Q^D_{i,j}$;
7          calculate $Q^L_{i,j}$, $Q^M_{i,j}$ and $\mathcal{A}'_i$;
8          if $Q_i(\mathcal{A}'_i, \mathcal{A}_{-i}) \le Q_i(\mathcal{A}_i, \mathcal{A}_{-i})$ then
9              send strategy update request to the BS;
10         else
11             keep the old strategy $\mathcal{A}_i$;
12     STAGE 3: BS receives all requests from MDs;
13     if there are update requests from MDs then
14         select the winner;
15         if the new strategy is D2D offloading then
16             if constraint (9), (10) is not satisfied then
17                 select another winner;
18         send confirmation signal to the winner;
19     else
20         the game reaches to NE;
21     STAGE 4: MDs wait for the messages from BS and
        other MDs;
22     if MD $z$ wins then
23         broadcast strategy update message to other MDs;
24         update strategy to $\mathcal{A}'_z$;
25     else
26         keep the old strategy $\mathcal{A}_z$;
27     $t \leftarrow t + 1$;
Output: The NE strategy $\mathcal{A}^*_i$ of each MD $i$.
```

Next, we describe DM-DOA that is shown in Algorithm 1 from the following five stages.

1) *STAGE 1 (Decision Making):*
   a) The idle MDs reports the available resource to the BS according to their willingness, then the BS integrates and sends the availability of MHs and the current task queue of server to each MD that has computation task periodically.
   b) For a busy MD i, it collects enough information about the available MHs and servers. In the meanwhile, it monitors if there is other MD updating strategy.
   c) MD i calculates $Q^L_{i,j}$, $Q^M_{i,j}$, $Q^D_{i,j}$, and selects the strategy $\mathcal{A}'_i$ with lowest cost.
2) *STAGE 2 (Strategy Updating):*
   a) Based on the current strategies of other MDs $\mathcal{A}_{-i}$, MD i calculates the cost function $Q_i(\mathcal{A}_i, \mathcal{A}_{-i})$ and $Q_i(\mathcal{A}'_i, \mathcal{A}_{-i})$ for the old strategy $\mathcal{A}_i$ and the new strategy $\mathcal{A}'_i$.
   b) If $Q_i(\mathcal{A}'_i, \mathcal{A}_{-i}) \leq Q_i(\mathcal{A}_i, \mathcal{A}_{-i})$, send the new strategy $\mathcal{A}'_i$ to the BS to contend the update chance, and otherwise.
3) *STAGE 3 (Resource Contention):*
   a) In each time slot, the BS decides which MD is the winner in this round of contention according to the contention principle.
   b) The BS sends a confirmation signal to the winner.
4) *STAGE 4 (Result Confirmation):*
   a) The winner that receives the confirmation signal updates its strategy and broadcasts the strategy update message to other MDs.
   b) The other MDs keep the old strategy and repeat STATE 1 and 2 for the next time slot.
5) *STAGE 5 (Task Performing):*
   a) The contention game converges to an NE when the BS does not receive strategy update request during a number of time slots.
   b) If no strategy update message is broadcast in the wireless channel, the MDs perform their tasks according to the last strategy.

The contention game will reach to an NE if MD has no motivation to change strategy and it keeps the current strategy $\mathcal{A}_i$ for the coming time slot $t+1$. Furthermore, the algorithm DM-DOA motivated by the FIP of game Γ, ensures that the asynchronous process of strategy updating reaches NE after a finite number of iterations. In practical system, the process of strategy updating will terminate when there is no update message broadcast during a number of time slots, then MDs can execute the computation tasks according to the strategies achieved in the last time slot.

3) Performance Analysis: The algorithm DM-DOA is more practical than EBOA, however, we should admit that there may be more than one NE. The proposed DM-DOA helps the game reach to one of them randomly, which may not be the best one. For analyzing the complexity of the algorithm and quantify the deviation of the performance in a reached NE from the performance in the optimal state, we discuss the computation complexity and the PoA in this section.

First, the computation complexity of DM-DOA is determined by the iterations. In each iteration, the MDs that have task execute STAGE 1-2 locally and exchange information with each other, which is basic arithmetical operation. Assuming that the system converges to NE after K iterations, the total computational complexity is $O(K * (M - N))$. Furthermore, to evaluate the performance of the reached NE, we introduce the PoA that is defined as the ratio of the overall cost in the worst NE and the overall cost in the optimal state of the system, which is shown as follows:

$$PoA = \frac{\max \sum_{\{i\in\mathcal{M}\}\backslash\{i\in\mathcal{N}\}} \left(Q_i\left(\mathcal{A}^*_i\right)\right)}{\min \sum_{\{i\in\mathcal{M}\}\backslash\{i\in\mathcal{N}\}} \left(Q_i(\mathcal{A}_i)\right)}. \tag{16}$$

According to (16), it is obviously that the lower bound for *PoA* is 1, hence we derive an upper bound. In DM-DOA, the D2D offloading is determined by the willingness of MH and the cooperation constraint. It causes a bad case: For any MD i that has task to perform, it can not establish D2D offloading due to the limitation of cooperation constraint, even though there are enough idle devices that are willing to provide computation support. In this case, the MD that has lowest cost in D2D offloading need exchange strategy to MEC offloading or local computing. If they choose MEC offloading, the cost of the MDs that offload to MEC server will increase due to the potential waiting time. Recalling (2), the computation delay in MEC server will be influenced by others' strategies. If the MEC offloading cost is higher than local computing, the MD will exchange to local computing. Hence, the worst case of NE is all MDs choose local computing, and the numerator of *PoA* is $\sum_{\{i\in\mathcal{M}\}\backslash\{i\in\mathcal{N}\}} Q^L_{i,j}$, where $j = 0$.

Furthermore, we derive a lower bound of the denominator of PoA. In the optimal state of the system, each MD will choose the lowest cost in the three offloading strategies. For MEC offloading, the best cast is that there is no competitor (i.e., the MEC offloading cost can be rewritten as $\bar{Q}^M_{i,j} = \alpha_i \cdot ([D_i/r^M_{i,j}] + [C_i/f^M_j]) + \beta_i \cdot P_i \cdot [D_i/r^M_{i,j}])$. We use $\bar{Q}_i(\mathcal{A}_i)$ to represent the modified cost, then the upper bound of PoA can be shown as

$$PoA \leq \frac{\sum_{\{i\in\mathcal{M}\}\backslash\{i\in\mathcal{N}\}} Q^L_{i,0}}{\sum_{\{i\in\mathcal{M}\}\backslash\{i\in\mathcal{N}\}} \min_{\mathcal{A}_i\in\mathcal{A}} \bar{Q}_i(\mathcal{A}_i)}. \tag{17}$$

V. Performance Evaluation and Discussion

In this section, we evaluate the performance of our proposed DM-DOA in terms of offloading delay, energy consumption, scalability, convergence, and the adaption to dynamic resource allocation.

A. Parameter Setting

In an ultradense network, the base station deployment density is defined as $\leq 10^3$ cells/km^2 [35], and the upper bound of the active user density in a dense urban scenario is 600 active users/km^2 [36]. Nevertheless, a desired network configuration (adequate access nodes and computing resources) is difficult to achieve due to the modification overhead. We consider a dense MEC network in a 300 m $\times$ 300 m region, where nine BSs are deployed in equidistant distribution and MDs are distributed randomly. Moreover, we randomly generate one third of the idle devices from the total MDs for initiation. For each task,

TABLE II
SIMULATION PARAMETER DESCRIPTION AND SETUP

Parameter	**Value and unit**
Number of MDs M	$[20, 200]$
Number of BSs B	9
Number of idle devices N	30% of M
Input data size D_i	$500 - 2000$ KB
Output data size O_i	a half of D_i
Task processing density γ	3 Megacycles/KB
Required CPU cycles C_i	$1500 - 6000$ Megacycles
Computing capacity of MD f_i^M	$0.8 - 1.6$ GHz
Computing capacity of server f_j^S	$4 - 8$ GHz
Transmission power of MD P_i	100 mW
Sub-channel bandwidth W	0.5 MHz
Noise power N_0	-100 dBm
Pass loss exponent	4
Energy coefficientε	500 mJ/Gigacycle
Delay coefficient α_i	$[0, 1]$ where $\alpha_i + \beta_i = 1$
Energy consumption coefficient β_i	$[0, 1]$ where $\alpha_i + \beta_i = 1$

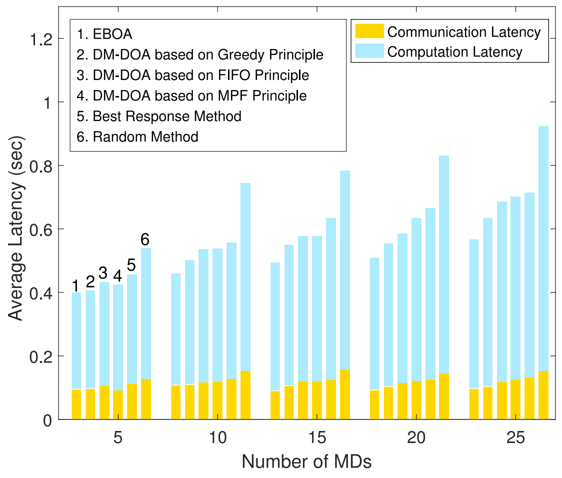
we set the input data size between 500 and 2000 kB, then let the output data size be a half of the input data size according to [24] and [37]. The number of required CPU cycles for each task varies from 1500 to 6000 megacycles and the acceptable latency is determined by the local execution time. Table II lists more details of other parameters in our simulation.

Besides the enumeration-based offloading method, we introduce other two methods in the same parameter setting for comparison.

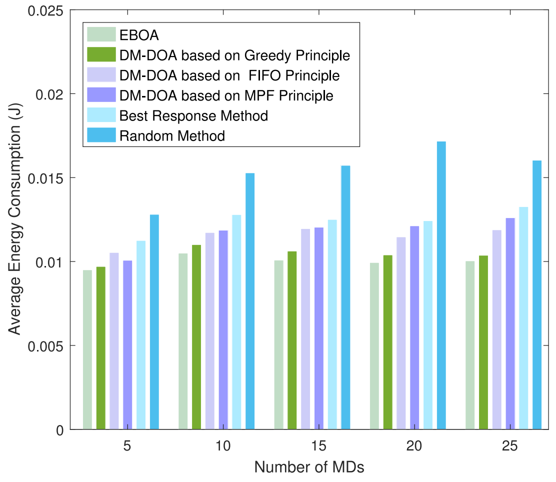
1) *Random Method:* The BS randomly sorts all MDs that have tasks to offload, then let them to choose their feasible strategies sequentially. Note that the feasible strategy is selected randomly and compared with the local computing cost, which may not be the best step. The random method is a straightforward method with low complexity, but may not help the system realize better performance.
2) *Best Response Method:* The best response strategy proposed in [28] also works in an iterative manner, where each user updates the strategy according to the computation delay (which may be influenced by the current task execution queue) in each server (referred as fog node).

Next, we give the evaluation metrics in our simulation. In addition to some intuitive metrics such as average communication delay, average computation delay, average energy consumption, iterations, and the overall cost of the system, we further introduce performance gain, energy saving ratio, and offloading ratio, which can be calculated as follows.

1) *Offloading Utility [19]:* It is the difference between the overall cost reached by the offloading algorithm and the local computing cost for all MDs.
2) *Performance Gain:* It is the ratio of the offloading utility to the local computing cost.
3) *Offloading Ratio:* It reflects the proportion of each strategy in all strategies (i.e., local computing, MEC offloading and D2D offloading) when the system is stable (e.g., reach to an NE).



(a)

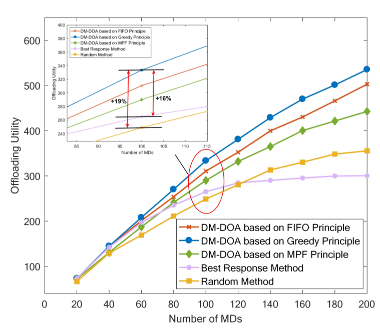


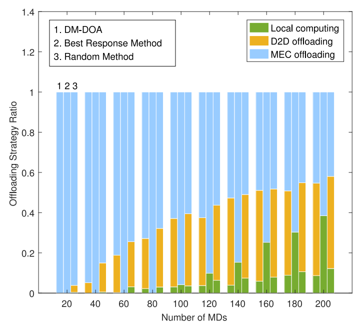
(b)

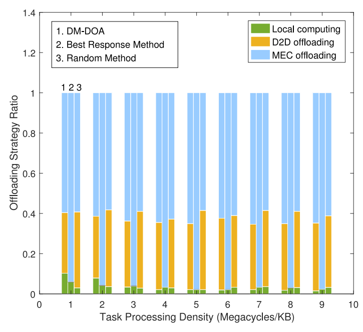
Fig. 3. Comparisons of (a) average delay and (b) average energy consumption between DM-DOA and the baseline algorithms when $M \in [0, 25]$ and $S = 9$.

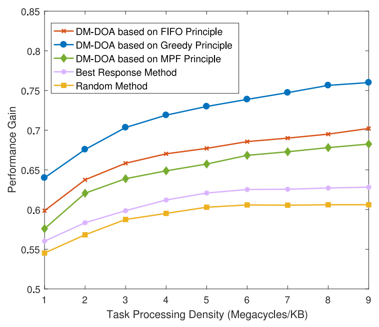
B. Numerical Results

1) Offloading Delay and Energy Consumption: To investigate the offloading delay and the energy consumption of our proposed DM-DOA, we compare the the proposed DM-DOA based on different contention principles with other three baseline methods (i.e., EBOA, best response method, and random method). As shown in Fig. 3, the average delay and energy consumption increase with the number of MDs. The algorithm EBOA can help the system find a globally optimal offloading strategy set, which brings a minimum delay and energy consumption. The average delay and energy consumption in the proposed DM-DOA is slightly higher than that in EBOA, which performs better than random method and best response method. Specifically, the computation delay accounts for larger part of the total delay, which demonstrates that an offloading strategy is high-efficiency comparing with local computing in a normal communication condition. Moreover, in the three contention principles, the greedy principle brings lower delay and energy


(a)


(b)


(c)


(d)

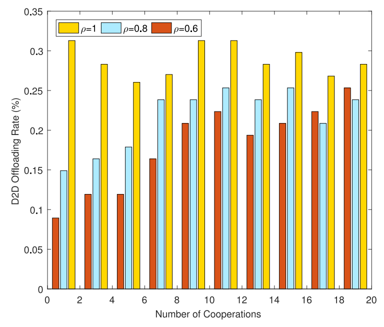

(e)

Fig. 4. Scalability evaluation in different user amount and task processing density. (a) Offloading utility, where $S = 9$ and $\gamma = 3$. (b) Offloading strategy ratio, where $S = 9$ and $\gamma = 3$. (c) Offloading strategy ratio, where $S = 9$ and $MD = 100$. (d) Performance gain, where $S = 9$ and $MD = 100$. (e) D2D offloading rate, where $M = 100$, $S = 9$ and $\gamma = 3$.

consumption comparing with other principles. This is because other principles select the winner according to the data rate or the potential function, rather than the offloading cost.

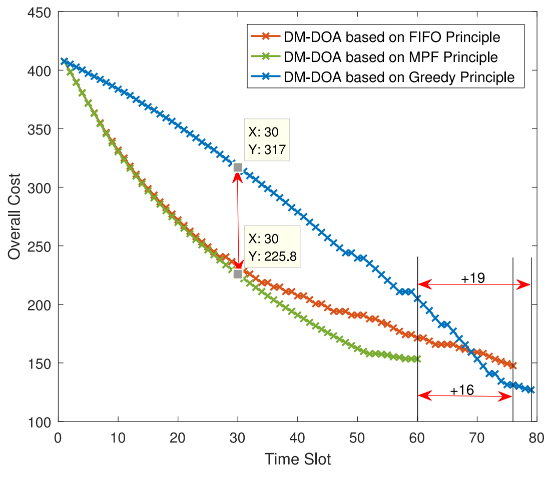
2) Scalability: To evaluate the scalability, we count the offloading utility and the offloading strategy ratio in different user amount. As shown in Fig. 4(a), the utility in DM-DOA based on greedy principle is 19% higher than random method, and 16% higher than best response method when $M = 100$. As the number of MDs increases, the difference becomes more significant. Note that the idle resource is not considered in best response method, which causes users prefer to local computing when there are more competitors. Hence, the utility tends to be stable and is lower than random method. For evaluating the strategy selection of MDs, we count the offloading strategy ratio of DM-DOA based on FIFO principle, best response method, and random method. As illustrated in Fig. 4(b), More users prefer an offloading strategy rather than local computing due to that the computation delay accounts for larger part of the total delay, and our algorithm shows higher offloading ratio.

In order to evaluate the impact of task processing density, we let the task processing density γ increase from 1 to 9, and count the number of MDs for different strategies as shown in Fig. 4(c). For DM-DOA and best response method, more users choose local computing if the processing density is not large, while the offloading strategy ratio is stable for random method. Furthermore, we calculate the performance gain versus task processing density as shown in Fig. 4(d). Obviously, DM-DOA has higher performance gain due to the increasing offloading utility. Additionally, for the compute-intensive task, random method, and best response method perform worse due to that the idle resource is not considered and the individual gain is not guaranteed.

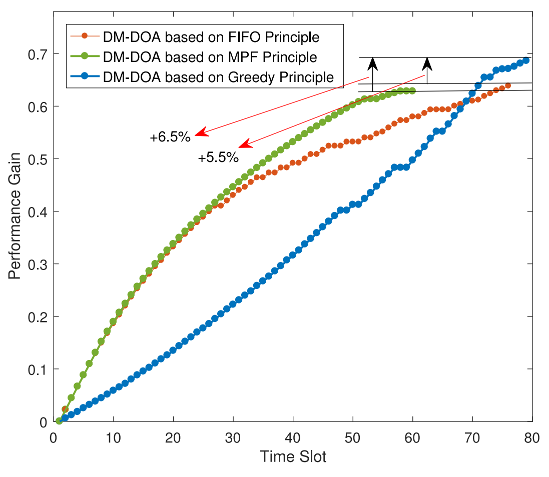
To evaluate the offloading rate that influenced by the cooperation credit, we count the D2D offloading rate when the resource contention game reaches to an NE, which means one cooperation is finished among users. Recalling (7), we can calculate the upper bound $\mathrm{Con}^{\max}$ for $\mathrm{Con}_j(k)$ according to C_i and f_j^M. Let $\rho = \overline{\mathrm{Con}}/\mathrm{Con}^{\max}$, we can count the D2D offloading rate for different contribution threshold $\overline{\mathrm{Con}}$ by changing the value of ρ.[3] Obviously, a higher value of ρ means a looser contribution threshold, and MDs will obtain more resource for free. Note that we randomly adjust the task state of each MD (i.e., if it has task to perform) but keep the number of idle devices invariable. As shown in Fig. 4(e), the D2D offloading rate fluctuates between 0.25 and 0.3 when $\rho = 1$, which means the idle devices are generous to contribute resource. However, the D2D offloading rate is lower when $\rho = 0.8$ and 0.6 due to the strict cooperation constraint. As the number of cooperations increases, it will increase close to the case of $\rho = 1$.

3) Impact of Contention Principle: To show the impact of different contention principle, we observes the overall cost the performance gain for FIFO principle, greedy principle and

[3]In our simulation, to avoid the influence of cooperation constraint on the evaluations of other metrics, we set $\rho = 1$ for other experiments.

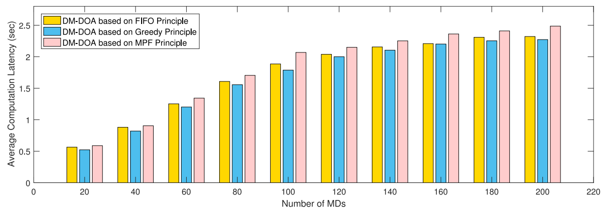


(a)

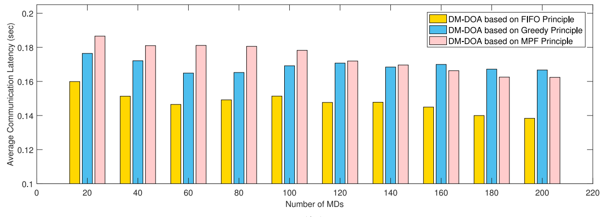


(b)

Fig. 5. Comparisons of overall cost and performance gain in three contention principles when $M = 100$, $S = 9$, and $\gamma = 3$. (a) Overall cost in each time slot. (b) Performance gain in each time slot.



(a)



(b)

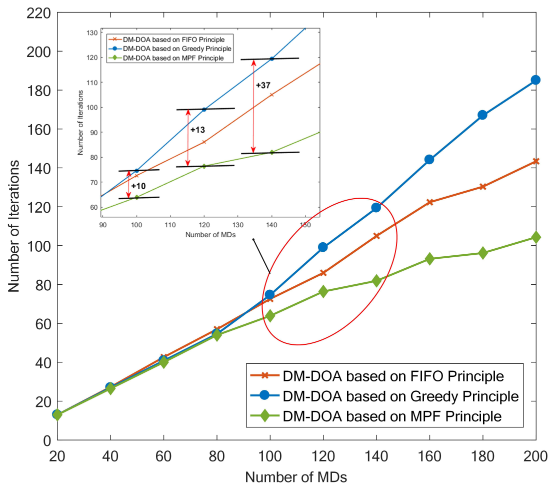
Fig. 6. Comparisons of average communication delay and average computation delay in three contention principles when $S = 9$ and $\gamma = 3$.

MPF principle, respectively. As shown in Fig. 5(a), the greedy principle brings the lowest overall cost, however, it requires more time slots to reach to NE. On the contrary, the MPF principle can help the system reach to NE faster, but the overall cost in NE state may not be the lowest, which is even higher than FIFO. In MPF principle, the BS selects the winner whose offloading cost can minimize the potential function, hence the overall cost decreases faster in the iteration process. Specifically, in the 30th time slot, the overall cost decreases to two-thirds of that in greedy principle. Similar tendency is appeared in performance gain evaluation in Fig. 5(b).

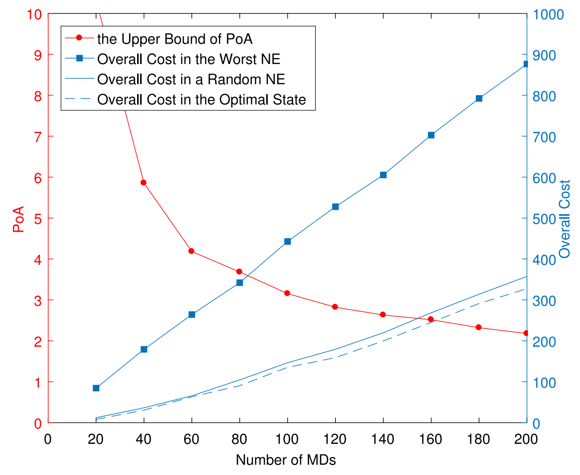
Moreover, the average computation delay and the average communication delay are counted in Fig. 6. There is not significant difference in computation delay, while in the perspective of communication delay, FIFI principle shows greater advantage. According to the first-in–first-out principle, the BS selects the winner who arrives first, which indicates the user with better communication condition has higher probability to be selected. Therefore, the average communication delay in FIFO principle is the lowest.

4) Convergence Behavior: We next investigate the convergence behavior of the proposed algorithm DM-DOA. The iterations is given in Fig. 7(a), where the proposed DM-DOA based on MPF principle requires less iterations. To measure that if the number of iterations is acceptable in a latency-sensitive scenario, we map the slotted time to the actual time. We allocate 100 bytes for the interaction message, which is enough to store the message of a single interaction between MDs or MDs and BSs. We further define t as the time cost of sending a 100-byte message, and derive that $t \leq 0.4$ ms according the transmission rate. Recall the five stages of DM-DOA that contains five interactions: 1) the idle MDs report available resource; 2) the BS broadcasts the resource availability; 3) MDs send strategy update requests; 4) the BS send confirmation signal to the winner; and 5) the winner broadcasts the new strategy, we obtain that the time cost is $5t$ in each round of contention. According to Fig. 7(a), the number of iterations of DM-DOA is less than 200 when $M = 200$. Hence we can obtain that, the time cost to reach to NE is less than 400 ms, which can be acceptable in the real world. Moreover, the upper bound of PoA is shown in Fig. 7(b), which declines rapidly and tends to the ideal value 1 as the number of MDs increases. We also show the overall cost in the worst NE and a random NE, and we give the overall cost in the optimal state for comparison. As the number of MDs increases, there are little difference between the optimal state and the random NE, and the performance loss of DM-DOA is less than 10% in different number of MDs. This indicates that the selfish behavior of users in DM-DOA has little effect on the overall offloading cost.

5) Dynamic Resource Adaption: Recalling the parameter setting, we generate 1/3 MDs as idle devices randomly for initiation. These devices are not always idle and willing to contribute resource, which causes the D2D offloading fails. Moreover, the establishment of D2D offloading also depends

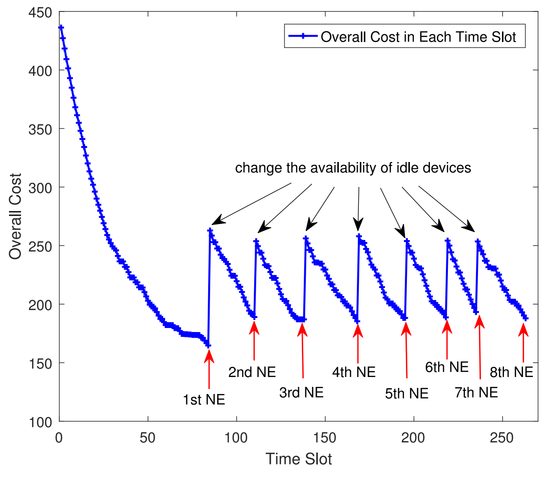


(a)

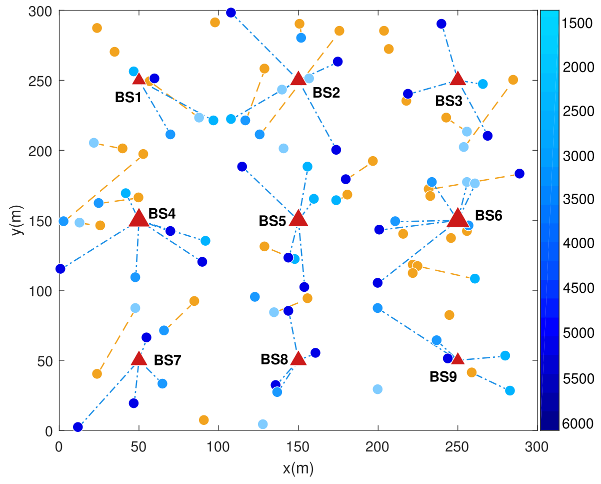


(b)

Fig. 7. Convergence behavior evaluation, where $S = 9$ and $\gamma = 3$. (a) Number of iterations. (b) The upper bound of PoA and the overall cost in different system states.



(a)



(b)

Fig. 8. Dynamic resource adaption evaluation, where $M = 100$, $S = 9$, and $\gamma = 3$. (a) The overall cost in different NE. (b) The connection in the first NE in (a).

on the energy constraint in (10). The above features require our algorithm to adapt to the dynamic resource allocation, hence we randomly change the availability of a number of idle devices from the original $1/3$ MDs, then record the overall cost for each iteration and the number of time slots when the NE reached. As shown in Fig. 8(a), the first NE is reached after 130 time slots approximately. Next, the availability of a number of idle devices is changed, and the MD that chooses D2D offloading may need to choose another strategy. Nevertheless, a new NE will be achieved after dozens of time slots, which illustrates the dynamic resource adaption of the proposed algorithm. Note that we change the availability of the idle devices from the original $1/3$ MDs in the simulation, which causes the amount of resource in the later NE is less than it in the first NE, and the overall cost increases but stays in a stable level. We further give the connection of NE state in Fig. 8(b), where the red triangle, blue solid circle and the yellow solid circle represent the BS, the busy MD, and the idle MD, respectively. The size of the triangle indicates the level of the computation capability of the MEC server, and the color depth of the blue solid circle represents the level of the computation density of each task in MDs (i.e., the required CPU cycles of each task increase from 1500 Megacycles to 6000 Megacycles). We find that the MDs that have compute-intensive tasks (i.e., the dark blue solid circle) prefer to choose MEC offloading, unless there are many competitors around. Moreover, if there is abundant resource in the MEC server (e.g., such as BS6), more MDs prefer to offload to MEC server, although there are available idle devices nearby. If the computation capability of MEC server is not enough (e.g., such as BS2), users will try to find D2D links for offloading. However, if the MD is far away from the BS and the task is not compute-intensive, it will choose local computing (see the MDs near the BS8).

C. Discussion

Different from the traditional MEC architecture, we focus to research the D2D-enabled offloading problem in a multiusers network with dynamic resource. Specially, we devote to satisfy the QoS of users and respect their individual behaviors, meanwhile, minimize the overall cost in a distributed manner. Comparing with the previous works, the proposed cooperative offloading is based on the assistance of the base station, which performs the idle device discovery and manages the D2D offloading process. In the cooperative offloading process, the strategy update and result confirmation only require simple messages with a little signaling overhead, and the time cost of the resource contention process is acceptable in the real world.

For future work, we schedule to improve the communication model, where the small scale fading will be considered. Moreover, for high mobility scenarios, we will focus on the connection switching between users and base stations, which influences the results return.

VI. Conclusion

In this article, we studied the D2D-enabled MEC computation offloading problem and proposed a game theoretic computation offloading scheme for satisfying the high QoS requirements of users and adapting the dynamic available resource in idle devices. By regarding the offloading process as a resource contention game, we developed an offloading algorithm DM-DOA for MDs to contend for computation resource. In particular, we provided game-theoretic analysis for the computation offloading problem under MEC and D2D environment. Numerical results demonstrated that the proposed algorithm performed well in terms of scalability, convergence, and dynamic resource adaption.

Appendix
Proof of Theorem 1

Given the potential function as shown in (14), now we prove that it satisfies (13) when any MD changes its strategy in four cases as follows.

1) *Local Computing → MEC Offloading:* For any MD i, if the offloading object is server j ($j \in \mathcal{S}$) under the new strategy $\mathcal{A}_i'$, we have

$$\begin{aligned}
&P(\mathcal{A}_i', \mathcal{A}_{-i}) - P(\mathcal{A}_i, \mathcal{A}_{-i})\\
&= \frac{1}{f_j^S} - \frac{1}{f_i^M} + \mathbf{G}_{i,j}(\mathcal{A}_i', \mathcal{A}_{-i}) - \mathbf{L}_i\\
&= \frac{1}{f_j^S} - \frac{1}{f_i^M} + \left(\frac{D_i}{C_i \cdot r_{i,j}^M} + \frac{\sum\limits_{i\in\mathcal{M}} \pi_{i,j}}{\mu_j \cdot c_i} + \frac{\beta_i}{\alpha_i} \cdot \frac{P_i \cdot D_i}{C_i \cdot r_{i,j}^M}\right)\\
&\quad - \frac{\beta_i}{\alpha_i} \cdot \varepsilon\\
&= \left(\frac{1}{f_j^S} + \frac{D_i}{C_i \cdot r_{i,j}^M} + \frac{\sum_{i\in\mathcal{M}} \pi_{i,j}/c_i}{\mu_j}\right.\\
&\qquad \left. + \frac{\beta_i}{\alpha_i} \cdot P_i \cdot \frac{D_i}{C_i \cdot r_{i,j}^M}\right) - \left(\frac{1}{f_i^M} + \frac{\beta_i}{\alpha_i} \cdot \varepsilon\right)\\
&= \frac{1}{w_i \cdot C_i} \cdot \left(Q_{i,j}^M(\mathcal{A}_i, \mathcal{A}_{-i}) - Q_{i,0}^L\right)\\
&= \frac{1}{w_i \cdot C_i} \cdot \left(Q_i(\mathcal{A}_i', \mathcal{A}_{-i}) - Q_i(\mathcal{A}_i, \mathcal{A}_{-i})\right).
\end{aligned} \tag{18}$$

If MD i change the strategy from MEC offloading to local computing, which means that $Q_{i,j}^M(\mathcal{A}_i, \mathcal{A}_{-i}) \geq Q_{i,0}^L$. The proof is similar to (18).

2) *Local Computing → D2D Offloading:* We assume that MD i offloads the task to an idle MH j ($j \in \bar{\mathcal{M}}$), then we have

$$\begin{aligned}
P(\mathcal{A}_i', \mathcal{A}_{-i}) - P(\mathcal{A}_i, \mathcal{A}_{-i}) &= \frac{1}{f_j^M} - \frac{1}{f_i^M} + \mathbf{R}_{i,j} - \mathbf{L}_i\\
&= \frac{1}{f_j^M} - \frac{1}{f_i^M} + \left(\frac{D_i}{C_i \cdot r_{i,j}^D} + \frac{Q_i}{C_i \cdot r_{j,i}^D} + \frac{\beta_i}{\alpha_i} \cdot \frac{P_i \cdot D_i}{C_i \cdot r_{i,j}^D}\right)\\
&\quad - \frac{\beta_i}{\alpha_i} \cdot \varepsilon\\
&= \left(\frac{1}{f_j^M} + \frac{D_i}{C_i \cdot r_{i,j}^D} + \frac{Q_i}{C_i \cdot r_{j,i}^D} + \frac{\beta_i}{\alpha_i} \cdot P_i \cdot \frac{D_i}{C_i \cdot r_{i,j}^D}\right)\\
&\quad - \left(\frac{1}{f_i^M} + \frac{\beta_i}{\alpha_i} \cdot \varepsilon\right)\\
&= \frac{1}{w_i \cdot C_i} \cdot \left(Q_{i,j}^D(\mathcal{A}_i, \mathcal{A}_{-i}) - Q_{i,0}^L\right)\\
&= \frac{1}{w_i \cdot C_i} \cdot \left(Q_i(\mathcal{A}_i', \mathcal{A}_{-i}) - Q_i(\mathcal{A}_i, \mathcal{A}_{-i})\right).
\end{aligned} \tag{19}$$

For the case that the strategy of MD i changes from D2D offloading to local computing, the proof is similar to (19).

3) *MEC Computing → D2D Offloading:* We assume that offloading object changes from server j ($j \in \mathcal{S}$) to an idle MH j' ($j' \in \bar{\mathcal{M}}$), then we have

$$\begin{aligned}
&P(\mathcal{A}_i', \mathcal{A}_{-i}) - P(\mathcal{A}_i, \mathcal{A}_{-i})\\
&= \frac{1}{f_{j'}^M} - \frac{1}{f_j^S} + \mathbf{R}_{i,j'} - \mathbf{G}_{i,j}(\mathcal{A}_i', \mathcal{A}_{-i})\\
&= \frac{1}{f_{j'}^M} - \frac{1}{f_j^S} + \left(\frac{D_i}{C_i \cdot r_{i,j'}^D} + \frac{Q_i}{C_i \cdot r_{j',i}^D} + \frac{\beta_i}{\alpha_i} \cdot \frac{P_i \cdot D_i}{C_i \cdot r_{i,j'}^D}\right)\\
&\quad - \left(\frac{D_i}{C_i \cdot r_{i,j}^M} + \frac{\sum_{i\in\mathcal{M}} \pi_{i,j}}{\mu_j \cdot c_i} + \frac{\beta_i}{\alpha_i} \cdot \frac{P_i \cdot D_i}{C_i \cdot r_{i,j}^M}\right)\\
&= \left(\frac{1}{f_{j'}^M} + \frac{D_i}{C_i \cdot r_{i,j'}^D} + \frac{Q_i}{C_i \cdot r_{j',i}^D} + \frac{\beta_i}{\alpha_i} \cdot P_i \cdot \frac{D_i}{C_i \cdot r_{i,j'}^D}\right)\\
&\quad - \left(\frac{1}{f_j^S} + \frac{D_i}{C_i \cdot r_{i,j}^M} + \frac{\sum_{i\in\mathcal{M}} \pi_{i,j}/c_i}{\mu_j}\right.\\
&\qquad\qquad \left. + \frac{\beta_i}{\alpha_i} \cdot P_i \cdot \frac{D_i}{C_i \cdot r_{i,j}^M}\right)\\
&= \frac{1}{w_i \cdot C_i} \cdot \left(Q_{i,j}^D - Q_{i,j}^M(\mathcal{A}_i, \mathcal{A}_{-i})\right)\\
&= \frac{1}{w_i \cdot C_i} \cdot \left(Q_i(\mathcal{A}_i', \mathcal{A}_{-i}) - Q_i(\mathcal{A}_i, \mathcal{A}_{-i})\right).
\end{aligned} \tag{20}$$

When the strategy of MD i changes from D2D offloading to MEC offloading, the proof shown above also applies.

4) *Exchange D2D Offloading Object:* We assume that offloading object changes from an MH j ($j \in \bar{\mathcal{M}}$) to

another MH j' ($j' \in \bar{\mathcal{M}}$), then we have

$$\begin{aligned}
P(\mathcal{A}'_i, \mathcal{A}_{-i}) - P(\mathcal{A}_i, \mathcal{A}_{-i}) &= \frac{1}{f^M_{j'}} - \frac{1}{f^M_j} + \mathbf{R}_{i,j'} - \mathbf{R}_{i,j} \\
&= \frac{1}{f^M_{j'}} - \frac{1}{f^M_j} + \left(\frac{D_i}{C_i \cdot r^D_{i,j'}} + \frac{Q_i}{C_i \cdot r^D_{j',i}} + \frac{\beta_i}{\alpha_i} \cdot \frac{P_i \cdot D_i}{C_i \cdot r^D_{i,j'}}\right) \\
&\quad - \left(\frac{D_i}{C_i \cdot r^D_{i,j}} + \frac{Q_i}{C_i \cdot r^D_{j,i}} + \frac{\beta_i}{\alpha_i} \cdot \frac{P_i \cdot D_i}{C_i \cdot r^D_{i,j}}\right) \\
&= \left(\frac{1}{f^M_{j'}} + \frac{D_i}{C_i \cdot r^D_{i,j'}} + \frac{Q_i}{C_i \cdot r^D_{j',i}} + \frac{\beta_i}{\alpha_i} \cdot P_i \cdot \frac{D_i}{C_i \cdot r^D_{i,j'}}\right) \\
&\quad - \left(\frac{1}{f^M_j} + \frac{D_i}{C_i \cdot r^D_{i,j}} + \frac{Q_i}{C_i \cdot r^D_{j,i}} + \frac{\beta_i}{\alpha_i} \cdot P_i \cdot \frac{D_i}{C_i \cdot r^D_{i,j}}\right) \\
&= \frac{1}{w_i \cdot C_i} \cdot \left(Q^D_{i,j'} - Q^D_{i,j}\right) \\
&= \frac{1}{w_i \cdot C_i} \cdot \left(Q_i(\mathcal{A}'_i, \mathcal{A}_{-i}) - Q_i(\mathcal{A}_i, \mathcal{A}_{-i})\right). \qquad (21)
\end{aligned}$$

References

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

[2] L. M. Abualigah, *Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering*. Boston, MA, USA: Springer, 2019, pp. 1–7.

[3] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "Hybrid clustering analysis using improved krill herd algorithm," *Appl. Intell.*, vol. 48, no. 11, pp. 1573–7497, May 2018.

[4] L. Tang and S. He, "Multi-user computation offloading in mobile edge computing: A behavioral perspective," *IEEE Netw.*, vol. 32, no. 1, pp. 48–53, Jan./Feb. 2018.

[5] P. Bellavista, L. Foschini, and D. Scotece, "Converging mobile edge computing, fog computing, and iot quality requirements," in *Proc. IEEE FiCloud*, Prague, Czech, Aug. 2017, pp. 313–320.

[6] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1801–1819, 4th Quart., 2014.

[7] D. Van Le and C.-K. Tham, "Quality of service aware computation offloading in an ad-hoc mobile cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8890–8904, Sep. 2018.

[8] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 1, pp. 207–220, Jan. 2019.

[9] H. Guo, J. Liu, and H. Qin, "Collaborative mobile edge computation offloading for IoT over fiber-wireless networks," *IEEE Netw.*, vol. 32, no. 1, pp. 66–71, Jan. 2018.

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

[11] A. M. Khasawneh, O. Kaiwartya, A. Khalifeh, L. M. Abualigah, and J. Lloret, "Green computing in underwater wireless sensor networks pressure centric energy modeling," *IEEE Syst. J.*, vol. 14, no. 4, pp. 4735–4745, Dec. 2020.

[12] W. A. Jabbar, W. K. Saad, and M. Ismail, "MEQSA-OLSRV2: A multicriteria-based hybrid multipath protocol for energy-efficient and QoS-aware data routing in MANET-WSN convergence scenarios of IoT," *IEEE Access*, vol. 6, pp. 76546–76572, 2018.

[13] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[14] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[15] 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 *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 1285–1293.

[16] E. Hyytiä, T. Spyropoulos, and J. Ott, "Offload (only) the right jobs: Robust offloading using the Markov decision processes," in *Proc. IEEE WoWMoM*, Boston, MA, USA, Jun. 2015, pp. 1–9.

[17] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[18] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, "Efficient resource allocation for on-demand mobile-edge cloud computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8769–8780, Sep. 2018.

[19] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[20] J. Wen, C. Ren, and A. K. Sangaiah, "Energy-efficient device-to-device edge computing network: An approach offloading both traffic and computation," *IEEE Commun. Mag*, vol. 56, no. 9, pp. 96–102, Sep. 2018.

[21] L. Yang, H. Zhu, H. Wang, H. Qian, and Y. Yang, "Incentive propagation mechanism of computation offloading in fog-enabled D2D networks," in *Proc. IEEE DSP*, Shanghai, China, Nov. 2018, pp. 1–4.

[22] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in *Proc. IEEE ICC*, Paris, France, May 2017, pp. 1–6.

[23] Y. He, J. Ren, G. Yu, and Y. Cai, "D2d communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1750–1763, Mar. 2019.

[24] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

[25] S. Yu, R. Langar, and X. Wang, "A D2D-multicast based computation offloading framework for interactive applications," in *Proc. IEEE GLOBECOM*, Washington, DC USA, Dec. 2016, pp. 1–6.

[26] Y. Liu, S. Wang, J. Huang, and F. Yang, "A computation offloading algorithm based on game theory for vehicular edge networks," in *Proc. IEEE ICC*, Kansas City, MO, USA, May 2018, pp. 1–6.

[27] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. ACM MSWiM*, New York, NY, USA, Nov. 2015, pp. 271–278.

[28] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for IoT systems: A computation offloading game," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.

[29] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[30] G. Hu, Y. Jia, and Z. Chen, "Multi-user computation offloading with D2D for mobile edge computing," in *Proc. IEEE GLOBECOM*, Abu Dhabi, UAE, Dec. 2018, pp. 1–6.

[31] M. Kamel, W. Hamouda, and A. Youssef, "Ultra-dense networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2522–2545, 4th Quart., 2016.

[32] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *Proc. IEEE SOSE*, San Francisco, CA, USA, Mar. 2013, pp. 494–502.

[33] L. Tang and H. Chen, "Joint pricing and capacity planning in the IaaS cloud market," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 57–70, Jan.–Mar. 2017.

[34] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, 1996.

[35] M. Ding, D. Lopez-Perez, G. Mao, P. Wang, and Z. Lin, "Will the area spectral efficiency monotonically grow as small cells go dense?" in *Proc. IEEE GLOBECOM*, San Diego, CA, USA, Dec. 2015, pp. 1–7.

[36] J. Deissner and G. P. Fettweis, "A study on hierarchical cellular structures with inter-layer reuse in an enhanced GSM radio network," in *Proc. IEEE Int. Workshop Mobile Multimedia Commun.*, San Diego, CA, USA, 1999, pp. 243–251.

[37] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.

Yuhan Yang received the bachelor's degree in electronics and information engineering from Sichuan University, Chengdu, China, in 2018. She is currently pursuing the Ph.D. degree with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China.

Her current research interests include mobile edge computation, security in Internet of Things, and blockchain technology.

Chengnian Long (Senior Member, IEEE) received the B.S., M.S., and the Ph.D. degrees in control theory and engineering from Yanshan University, Hebei, China, in 1999, 2001, and 2004, respectively.

He was a Research Associate with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, and a Killam Postdoctoral Fellow with the University of Alberta, Edmonton, AB, Canada. Since 2009, he has been with Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he has been a Full Professor since 2011. His current research interests include artificial intelligence of things, blockchain technology, deep learning, and cyber–physical system security.

Jing Wu (Senior Member, IEEE) received the B.S. degree in electrical engineering from Nanchang University, Nanchang, China, in 2000, the M.S. degree in electrical engineering from Yanshan University, Hebei, China, in 2002, and the Ph.D. degree in electrical engineering from the University of Alberta, Edmonton, AB, Canada, in 2008.

Since 2011, she has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is currently an Associate Professor. Her current research interests include robust model predictive control, security control, and stability analysis, and estimations for cyber-physical systems.

Dr. Wu is a Registered Professional Engineer in Alberta, Canada.

Shaoliang Peng (Member, IEEE) received the B.S. degree in mathematics, and the M.S. and Ph.D. degrees in computer science and technology from National University of Defense Technology, Changsha, China, in 2001, 2003 and 2008, respectively.

He works with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, and the National University of Defense Technology, Changsha, and is an Adjunct Professor of BGI, Shenzhen, China. He was a Visiting Scholar with the CS Department, City University of Hong Kong, Hong Kong, from 2007 to 2008 and the BGI Hong Kong, Hong Kong, from 2013 to 2014. He is the Executive Director of the National Supercomputing Center, Changsha, and a Professor of the National University of Defense Technology. He has published dozens of academic papers on several internationally influential journals, including *Science*, *Nature Communications*, Cell AJHG, *Genome Biology*, *Cancer Research*, ACM/IEEE Transactions, and BIBM. His research interests are high performance computing, bioinformatics, big data, virtual screening, and biology simulation.

Bo Li (Fellow, IEEE) received the B.S. (*summa cum laude*) degree and M.S. degree in computer science from Tsinghua University, Beijing, China, in 1987 and 1989, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Massachusetts Amherst, Amherst, MA, USA, in 1994.

He is a Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. He has been the Chief Technical Advisor for Chinacache Corporation (a NASDAQ listed company), the leading CDN operator in China since 2008. From 2010 to 2013, he was a Cheung Kong Visiting Chair Professor with Shanghai Jiao Tong University, Shanghai, China, He was an Adjunct Researcher with Microsoft Research Asia from 1999 to 2007 and with Microsoft Advance Technology Center from 2007 to 2009. His current research interests include data-center networking, cloud computing, content distribution in the Internet, and mobile wireless networking.

Prof. Li received six best paper awards from the IEEE. He made pioneering contributions in the Internet video broadcast with a system called Coolstreaming, which was credited as first large-scale peer-to-peer live video streaming system in the world, that received the inaugural the Test-of-Time Paper Award from the IEEE INFOCOM in 2015. He received the Young Investigator Award from the Natural Science Foundation of China in 2005 and the State Natural Science Award (Second Class) in 2011. He has been an Editor or a Guest Editor for more than a two dozen of journals and magazines, mostly in IEEE and ACM. He was the Co-TPC Chair for IEEE INFOCOM 2004.