Task Execution Cost Minimization-Based Joint Computation Offloading and Resource Allocation for Cellular D2D MEC Systems

Rong Chai, *Member, IEEE*, Junliang Lin, Minglong Chen, and Qianbin Chen, *Senior Member, IEEE*

***Abstract*—The rapid proliferation of new applications poses great challenges to the computation capability of mobile devices. To tackle this problem, computation offloading was proposed as a promising paradigm. In this paper, we consider a cellular device-to-device (D2D) mobile edge computing (MEC) system consisting of one base station (BS) deployed with an MEC server and users. We assume that a portion of users have task computation requirements and may perform local computing, MEC offloading, or D2D offloading. Further assuming that tasks can be partitioned into small-sized data and can be executed simultaneously via various computation modes, we jointly study computation offloading and resource allocation problem. To achieve efficient information interaction and task management, we first propose a joint task management architecture. Defining task execution cost as the weighted sum of task execution latency and energy consumption, we then formulate the joint optimization problem as a task execution cost minimization problem. As the formulated problem is a mixed integer nonlinear problem, which cannot be solved conveniently, we propose a heuristic algorithm that successively solves computation offloading subproblem and resource allocation subproblem by Kuhn–Munkres algorithm and Lagrangian dual method, respectively. Numerical results demonstrate the effectiveness of the proposed algorithm.**

***Index Terms*—Cellular device-to-device (D2D) system, computation offloading, energy consumption, latency, resource allocation.**

I. Introduction

THE rapid development of mobile Internet and smart devices promotes the emergence of new applications such as interactive gaming, virtual reality, and augment reality [1]–[3]. As these applications or tasks may in general be resource-intensive and require high computation complexity, energy consumption, and short response time, executing tasks locally at mobile devices poses great challenges to the computation and process capability of the devices. To tackle this issue, mobile edge computing (MEC) technology was recently proposed as a promising paradigm of computation [4]. Through deploying high performance MEC servers at the edge of radio access networks, e.g., the base stations (BSs) of cellular networks, and performing computation offloading, i.e., allowing computation to be conducted at the MEC servers instead of mobile devices, the task execution latency can be reduced and the energy consumption of the mobile devices can be saved significantly.

In a computation offloading-enabled system, mobile devices may perform binary offloading, i.e., either executing task locally, also referred to as local computing or performing MEC offloading. To further enhance the performance of task execution, partial computation offloading scheme can be applied [5]. Under the assumption that one task can be partitioned into small portions with arbitrary data sizes, various portions of the original task can be executed independently in a local computing mode or in the manner of MEC offloading. By exploiting parallel computing at mobile devices and MEC servers, the latency required for executing the whole task can be reduced. Although MEC offloading offers the performance enhancement of task execution, the resource contention may occur at the MEC servers especially when a large number of tasks are offloaded to the servers. To address this problem, researchers propose to use device-to-device (D2D) communication technology for task offloading where a certain number of user equipment (UE) are capable of receiving the tasks from their neighboring UE via D2D transmission links and performing task execution on behalf of the neighboring UE [6], [7].

Although previous research works consider various offloading schemes and aim to design the optimal offloading strategies, the joint optimization of computation offloading modes with the resource allocation problem at the MEC servers failed to be studied extensively especially under the scenario that both MEC offloading and D2D offloading are allowed. Furthermore, although task execution latency or energy consumption has been stressed in previous works, the tradeoff between the two performance metrics was insufficiently explored. In this paper, we consider the computation offloading and resource allocation problem in a cellular D2D MEC system and propose a task execution cost minimization-based joint computation offloading and resource allocation algorithm.

The major contributions of this paper are as follows.

1) We design a joint task management architecture based on which the proposed joint computation offloading and resource allocation algorithm can be conducted.
2) The problem of computation offloading or resource allocation in MEC systems has been studied separately in previous research work [8]–[22]. In this paper, we jointly consider the computation offloading and resource

Manuscript received July 6, 2018; revised December 26, 2018, April 12, 2019, and May 30, 2019; accepted June 2, 2019. Date of publication June 28, 2019; date of current version November 22, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61571073 and in part by the Joint Scientific Research Fund of Ministry of Education and China Mobile under Grant MCM20160105. *(Corresponding author: Junliang Lin.)*

The authors are with the Key Lab of Mobile Communication Technology, Chongqing University of Posts and Telecommunications, Chongqing 40065, China (e-mail: chairong@cqupt.edu.cn; linjunliang1226@163.com; 229247063@qq.com; chenqb@cqupt.edu.cn).

Digital Object Identifier 10.1109/JSYST.2019.2921115

allocation problem in a cellular D2D MEC system where both MEC offloading and D2D offloading are enabled, and design jointly optimal strategies in order to achieve the optimal task execution performance.

3) Although Wang *et al.* [5], [23]–[28] considered the problem of joint computation offloading and resource allocation in an MEC-enabled system, they fail to jointly consider MEC offloading and D2D offloading modes, thus may result in highly limited task execution performance.
4) The task execution latency and energy consumption are both considered in a previous work, in this paper, we characterize the task execution cost as the weighted sum of task execution latency and energy consumption and formulate the joint computation offloading and resource allocation problem as a task execution cost minimization problem.
5) Since the formulated optimization problem is a mixed integer nonlinear optimization problem, which cannot be solved conveniently using traditional optimization tools, we propose a heuristic algorithm, which successively solves two subproblems, i.e., computation offloading subproblem and resource allocation subproblem, by means of Kuhn–Munkres algorithm and Lagrangian dual method, respectively.

II. Related Work

In this section, we present an overview of the related work, including computation offloading, resource allocation, and joint computation offloading and resource allocation in MEC systems.

A. Computation Offloading in MEC Systems

In past few years, computation offloading problem has been studied for MEC systems [8]–[15]. In [8] and [9], optimal computation offloading schemes were designed to minimize the task execution latency. Liu *et al.* [8] formulated the computation offloading problem as a Markov decision process and design an optimal power-constrained latency minimization-based computation offloading policy. Leveraging the idea of software-defined networking (SDN), Chen and Hao [9] addressed the computation offloading problem in an SDN-enabled ultra-dense network and proposed an optimal computation offloading algorithm, which minimizes the task execution latency.

Chen *et al.* [10]–[12] examined the energy consumption required for performing computation offloading. In [10], Yang *et al.* consider a small-cell MEC system and formulate the computation offloading problem as an energy consumption optimization problem in order to minimize the overall energy consumption of system-wide entities. Assuming that mobile devices are capable of harvesting renewable energy, Chen *et al.* [11] formulated the multiuser multitask computation offloading problem as an energy consumption minimization problem and proposed centralized and distributed greedy maximal scheduling algorithms to solve the problem. In [12], task execution latency and energy consumption are considered for a wireless system equipped with MEC servers, and an optimal frequency scaling and MEC server selection scheme is proposed to reduce both task execution latency and energy consumption.

The computation offloading overhead is considered in [13]–[15]. Chen *et al.* [13] formulated the computation offloading overhead minimization problem as an offloading game and demonstrated the existence of Nash equilibrium. In [14], a dynamic computation offloading framework is proposed and the computation offloading problem is formulated as a multilabel classification problem, which minimizes the computation and offloading overhead. In [15], Bi and Zhang formulated the joint computing mode selection and transmission time allocation problem in a multiuser MEC system as a weighted sum rate maximization problem and solve the problem by applying alternating direction method of multipliers (ADMM).

B. Resource Allocation in MEC Systems

Although the research work in [8]–[15] mainly focused on designing optimal computation offloading schemes for MEC systems, for a more general system scenario where multiple mobile devices may perform task offloading, the resource sharing and contention among the mobile devices may lead to system performance deterioration, which is highly undesired. In this context, optimal resource allocation (ORA) strategies should be designed [16]–[22].

Le *et al.* [16] considered a multiuser MEC system where each user is allocated a fraction of the radio resource of BS and the computation power of the MEC server. The resource allocation problem is formulated as a task execution latency minimization problem, which is then solved by means of a bisection-search method. The resource allocation problem in an orthogonal frequency division multiple access (OFDMA)-based MEC system is addressed in [17]. The joint subcarrier and power allocation problem is formulated to minimize the maximal task execution latency and a heuristic algorithm is proposed to obtain a low-complexity subcarrier assignment and power allocation strategy.

Stressing the energy consumption for task execution, Guo *et al.* [18] developed an energy-efficient resource allocation scheme for a multiuser MEC system. By optimally allocating the sequence and time duration for task execution, the weighted sum energy consumption can be minimized. In [19], Mao *et al.* studied the frequency scaling, transmit power, and bandwidth allocation problem in an MEC system. Under task buffer stability constraint, an energy consumption minimization problem is formulated and a Lyapunov optimization-based algorithm is developed to obtain the optimal computation offloading policy. To utilize system resources efficiently, Yu *et al.* [20] considered the radio and computation resource allocation issue in an OFDMA-based MEC system. An energy consumption minimization problem is formulated and a joint scheduling algorithm that allocates both subcarrier and central processing unit (CPU) frequency is proposed. In [21], by defining the system utility as the normalized weighted combination of the latency and energy consumption reduction, the problem of resource allocation is formulated as a system utility maximization problem and a low-complexity ranking-based algorithm is proposed to obtain the suboptimal computation offloading strategy. Considering the fairness of mobile users, Zhu *et al.* [22] proposed a fair Nash bargaining game-based resource allocation strategy to maximize the overall network throughput.

C. Joint Computation Offloading and Resource Allocation in MEC Systems

Although previous works in [8]–[22] mainly focus on designing either computation offloading scheme or resource allocation

scheme for MEC systems, it can be observed that resource allocation and computation offloading strategy may jointly affect the performance of task execution. Hence, it is highly desired to jointly design the two schemes for the MEC system in order to further enhance user quality of service or network utility, as have been demonstrated in some recent research work [5], [23]–[28].

Network revenue is formulated and maximized in [23] and [24] through optimally designing joint computation offloading and resource allocation schemes. Wang *et al.* [23] considered a content caching-enabled cellular MEC system and jointly consider computation offloading, resource allocation, and content caching strategies. The joint optimization problem is formulated as a network revenue maximization problem, which is solved by means of ADMM-based algorithm. In [24], He *et al.* proposed an integrated framework that enables dynamic orchestration of networking, caching, and computing resource for vehicular networks. A deep reinforcement learning process is proposed to characterize the joint optimization problem and the optimal strategy is obtained, which maximizes the long-term network revenue.

Energy consumption is considered as an important metric in designing joint computation offloading and resource allocation schemes [25]–[27]. In [25], the problem of computation offloading and subchannel allocation is considered for a multiuser MEC system and a joint optimal scheme is proposed to minimize the weighted sum energy consumption under the constraint on computation latency. Nguyen and Long [26] studied the joint computation offloading and subchannel allocation problem in a two-tier heterogeneous network. By formulating the joint optimal design problem as a maximum energy consumption minimization problem and solving the problem based on bisection search method, the optimal joint computation offloading and subchannel allocation strategy is obtained. In [27], Zhang *et al.* addressed the problem of joint computation offloading, subchannel allocation, and power allocation for heterogeneous networks enabling MEC offloading. An energy consumption minimization-based problem is formulated and a distributed joint optimization algorithm is proposed to achieve the optimal strategy.

Energy consumption and task execution latency are jointly considered in designing joint computation offloading and resource allocation strategies [5], [28]. In [5], Wang *et al.* designed a joint computation offloading, computation capacity, and power allocation framework, which aims to minimize the energy consumption of the mobile devices under latency constraint, and to minimize the task execution latency subject to energy constraint. Chen *et al.* [28] aimed to jointly optimize the computation offloading decisions and bandwidth allocation strategies for a multiuser computing system. By defining system cost as the weighted sum of energy consumption and task execution latency, the system cost minimization-based joint optimization problem is formulated as a nonconvex quadratically constrained quadratic program and separable semidefinite relaxation is applied to obtain the approximate solution.

Although previous research works exploit MEC offloading for task execution, they fail to consider D2D offloading mode, which may further enhance the performance of task execution [5], [8]–[28]. Pu *et al.* in [6] and [7] addressed the problem of D2D offloading in wireless networks. In [6], a D2D fogging framework was proposed where users have the flexibility to exploit the resource of each other. The task execution problem is formulated as an energy consumption minimization problem and an online task offloading algorithm is developed to obtain the optimal D2D offloading strategy. Chen and Zhang [7] considered a binary task offloading system where mobile users may choose local computing, D2D offloading, or MEC offloading for task execution and no parallel computing is allowed. By defining task execution cost as the weighted sum of latency and energy consumption, the problem of task offloading is formulated as a task execution cost minimization problem, which is solved by using the Edmonds's Blossom algorithm. However, Chen and Zhang [7] assume that the resources allocated to users are fixed, which cannot be assigned according to user requirements, hence may not achieve efficient resource utilization. Furthermore, binary offloading scheme fails to fully exploit the advantage of parallel computing, which may lead to potential performance degradation.

In this paper, we investigate the joint computation offloading and resource allocation problem in cellular D2D MEC systems. Assuming that various computation offloading modes including complete local computing, MEC offloading, and D2D offloading are enabled, and multiuser sharing on both communication and computation resources are allowed, we propose a task execution cost minimization-based joint computation offloading and resource allocation algorithm.

III. System Model and Proposed Joint Task Management Architecture

A. System Model

In this paper, we consider a cellular D2D MEC system, which consists of a number of mobile users and one BS deployed with an MEC server. We assume that a small portion of mobile users may need to execute computation task, for convenience, we refer to these users as request users (RUs). Denote N as the number of RUs and RU_i as the ith RU, $1 \leq i \leq N$. The computation task of RU_i can be characterized as $\xi_i \triangleq (I_i, D_i, T_i^{\max})$ where I_i (in bits) denotes the size of the input data of RU_i, D_i (in cycles) represents the amount of computation resource required to accomplish the task of RU_i and $T_i^{\max}$ is the maximum tolerable task execution latency of RU_i. For simplicity, we assume that all the tasks of the RUs are independent and can be executed independently.

We assume that the MEC server is capable of conducting computation offloading for the RUs, thus, instead of executing computation task locally, the RUs may offload their computation task to the MEC server via a cellular link. Furthermore, leveraging D2D communication technology, we assume that certain mobile users, which are of relatively high performance, would tend to offer D2D offloading service to their neighboring RUs. For simplicity, we refer the mobile users, which offer D2D offloading service for their D2D peers, as service users (SUs). Denote the jth SU as SU_j, $1 \leq j \leq M$, where M is the number of SUs. Similar to previous studies on computation offloading [29], [30], in this paper, we consider a quasistatic scenario where the characteristics of the system remain unchanged during a computation offloading period.

It is apparent that in the considered systems, the RUs may conduct task execution in three modes, i.e., local computing, MEC offloading, and D2D offloading. To enhance the performance of task execution, we apply parallel computing mechanism in this paper. More specifically, we assume that the tasks of RUs can

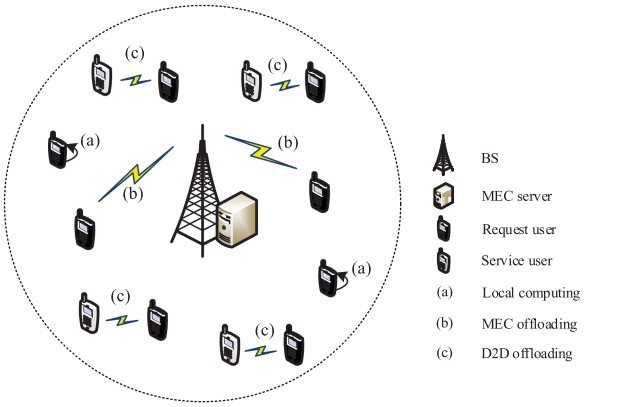

Fig. 1. System model.

be partitioned into small portions with arbitrary data size, and various portions of the task can be executed in a parallel manner in different offloading modes. In particular, we assume that parallel computing at the RUs and one of the offloading entities, i.e., the MEC server or one SU, is allowed. For simplicity, in this paper, we define MEC offloading mode as the computation scheme where parallel computing between local computing and MEC offloading are executed, or complete MEC offloading (i.e., no local computing) is executed, and define D2D offloading mode as the scheme where parallel computing between local computing and D2D offloading are executed, or complete D2D offloading (i.e., no local computing) is executed. We also define complete local computing mode as the computation scheme when the whole task is executed locally and no offloading is performed.

In this paper, we address the problem of computation offloading mode selection and the resource allocation strategy design for MEC offloading mode in cellular D2D MEC systems. Hence, we make a simplified assumption on bandwidth resource allocation. More specifically, we assume that various bandwidth resources are respectively allocated to the BS and the D2D peers to avoid the transmission interference between cellular links and D2D links. We further assume that the available bandwidth resource of the BS is divided into equal-width subchannels. Let W^{b} and L_0 denote, respectively, the amount of the bandwidth and the total number of subchannels of the BS. Hence, the bandwidth of each subchannel can be calculated as $W^{\mathrm{b}}_{\mathrm{sub}} = \frac{W^{\mathrm{b}}}{L_0}$. We assume that a number of subchannels of the BS can be allocated to RUs. To achieve efficient bandwidth resource utilization of the BS, we assume that multiple RUs are allowed to access the BS simultaneously, however, the RUs should occupy various subchannels of the BS. Moreover, to avoid the transmission interference between D2D links, we assume each D2D pair is assigned one orthogonal subchannel with the bandwidth of the subchannel being denoted by W^{d}.

To utilize the computation capability of the MEC server efficiently and enhance the task execution performance of the RUs as well, we assume that a number of RUs are allowed to offload their task to the MEC server simultaneously, in which case each RU will be allocated a fraction of the computation capability of the MEC server. Fig. 1 shows the system model considered in this paper.

B. Proposed Joint Task Management Architecture

To achieve the efficient information interaction and task management in the cellular D2D MEC system, we propose a joint task management architecture. Fig. 2 shows the proposed joint task management architecture, in which three functional entities, including global task management entity (GTME), local task management entity (LTME), and user task management entity (UTME), are introduced to tackle the dynamic information of the system and to conduct joint computation offloading and resource allocation for RUs. The major roles and functions of GTME, LTME, and UTME are as follows.

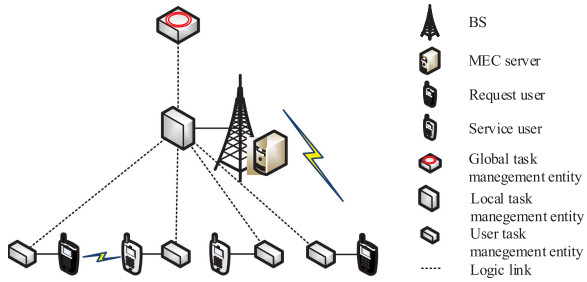

Fig. 2. Proposed joint task management architecture.

1) Global Task Management Entity: Being deployed over the BS, RUs, and SUs, GTME acts as the centralized controller of the system. Through interacting with the LTME, the GTME receives network and user state information and conducts the proposed joint computation offloading and resource allocation algorithm. The obtained optimal strategy will be sent back to the LTME.

2) Local Task Management Entity: LTME is deployed at the BS and acts as a local controller of the BS. Through interacting with the associated UTMEs and GTME, the LTME forwards user state information to the GTME, and forward the received joint computation offloading and resource allocation strategies to the UTMEs.

3) User Task Management Entity: Embedded in each RU and SU, UTME is responsible for collecting and storing channel state information, device characteristics and task requirements of users. Through interacting with the LTME, the UTME sends the collected information to the network and receives the joint computation offloading and resource allocation strategies.

IV. Optimization Problem Formulation

In this section, we define the task execution cost of the cellular D2D MEC system and formulate the joint computation offloading and resource allocation problem as a task execution cost minimization problem.

A. Objective Function

Stressing the performance of task execution of all RUs in the system, we define the total task execution cost U as

$$U = \sum_{i=1}^{N} U_i \tag{1}$$

where U_i denotes the task execution cost of RU_i. As RU_i may perform various computation offloading modes, the corresponding task execution cost U_i can be expressed as

$$U_i = x_i^0 U_i^0 + x_i U_i^{\mathrm{b}} + \sum_{j=1}^{M} x_{i,j} U_{i,j}^{\mathrm{d}} \tag{2}$$

where $x_i^0 \in \{0, 1\}$ denotes complete local computing variable, i.e., $x_i^0 = 1$, if RU_i is performing task execution in complete

local computing mode, otherwise, $x_i^0 = 0$; $x_i \in \{0, 1\}$ denotes the MEC offloading variable of RU_i, i.e., $x_i = 1$, if RU_i offloads its task to the MEC server, otherwise, $x_i = 0$; $x_{i,j} \in \{0, 1\}$ denotes the D2D offloading variable of RU_i, i.e., $x_{i,j} = 1$, if RU_i offloads its task to SU_j, otherwise, $x_{i,j} = 0$; U_i^0, U_i^{b}, and $U_{i,j}^{\mathrm{d}}$ denote, respectively, the task execution cost of RU_i in complete local computing mode, MEC offloading mode, and D2D offloading mode, $1 \le i \le N$, $1 \le j \le M$.

As both the task execution latency and the energy consumption required for task computation are important metrics characterizing the performance of task execution, in this paper, we define task execution cost as the weighted function of those two metrics. In the following sections, we formulate the task execution cost in various computation offloading modes.

1) Complete Local Computing Mode: The task execution cost of RU_i in complete local computing mode can be formulated as

$$U_i^0 = T_i^0 + \rho_i E_i^0 \tag{3}$$

where T_i^0 and E_i^0 denote respectively the task execution latency and the energy consumption required to complete the whole task computation locally, ρ_i is the scalar weight of RU_i. As ρ_i characterizes the relative importance of RU's preference on task execution latency and energy consumption, the slightly higher ρ_i reflects the more importance of energy consumption in comparison to the task execution latency. Referring to [31] and [32], we formulate T_i^0 as

$$T_i^0 = \frac{D_i}{F_i} \tag{4}$$

where F_i (in cycles/s) denotes the computation capability of RU_i, $1 \le i \le N$. In this paper, the computation capability of RUs, SUs, and the MEC server is defined as the frequency at which the CPU of the devices or the server is running, and thus characterizes the computational speed of the RUs, the SUs, and the MEC server.

E_i^0 in (3) can be calculated as [33]

$$E_i^0 = \delta_i D_i F_i^2 \tag{5}$$

where δ_i is the energy consumption coefficient related to the CPU performance of RU_i.

2) MEC Offloading Mode: In the case that RU_i performs MEC offloading, the task execution cost can be expressed as

$$U_i^{\mathrm{b}} = T_i^{\mathrm{b}} + \rho_i E_i^{\mathrm{b}} \tag{6}$$

where T_i^{b} and E_i^{b} denote, respectively, the task execution latency and energy consumption of RU_i in the MEC offloading mode.

Since parallel computing might be conducted in this case, the total task execution latency can be calculated as the longer latency required for local computing and MEC offloading, hence, we can express T_i^{b} as

$$T_i^{\mathrm{b}} = \max\left\{(1 - \lambda_i)T_i^0, \lambda_i T_i^{\mathrm{b},0}\right\} \tag{7}$$

where $\lambda_i \in [0, 1]$ denotes the task partition variable of RU_i in the MEC offloading mode, in particular, λ_i is defined as the fraction of RU_i's task being executed at the MEC server, and $1 - \lambda_i$ denotes the fraction of RU_i's task being executed locally in the MEC offloading mode, $T_i^{\mathrm{b},0}$ denotes the task execution latency required for performing the whole task at the MEC server, and can be calculated as

$$T_i^{\mathrm{b},0} = T_i^{\mathrm{b,t}} + T_i^{\mathrm{b,e}} \tag{8}$$

where $T_i^{\mathrm{b,t}}$ denotes the transmission latency required for RU_i to offload its task to the MEC server and $T_i^{\mathrm{b,e}}$ denotes the task computation latency at the MEC server. It is noticed that in calculating the total task execution latency, we ignore the transmission latency for a return link. The reason is that the amount of the output data of one task being processed at the MEC server is normally much smaller than that of the input data, hence, transmitting the output data from the MEC server to the RUs usually requires much smaller transmission latency, which can be ignored [13], [23].

$T_i^{\mathrm{b,t}}$ in (8) is given by

$$T_i^{\mathrm{b,t}} = \frac{I_i}{R_i} \tag{9}$$

where R_i denotes the achievable data rate of RU_i when transmitting to the MEC server and can be expressed as

$$R_i = \eta_i W_{\mathrm{sub}}^{\mathrm{b}} \log_2\left(1 + \frac{p_i^{\mathrm{b}} h_i}{\sigma^2}\right) \tag{10}$$

where $\eta_i \in \{0, 1, 2, \ldots, L_0\}$ denotes the number of subchannels allocated to RU_i when offloading to the MEC server, p_i^{b} denotes the transmit power of RU_i when transmitting to the BS, h_i and σ^2 denote, respectively, the channel gain and the noise power of the link between RU_i and the BS. For convenience, we define $\hat{\eta}_i = \frac{\eta_i}{L_0} \in \{0, \frac{1}{L_0}, \frac{2}{L_0}, \ldots, 1\}$ as the fraction of the bandwidth resource allocated to RU_i and rewrite (10) as

$$R_i = \hat{\eta}_i W^{\mathrm{b}} \log_2\left(1 + \frac{p_i^{\mathrm{b}} h_i}{\sigma^2}\right).$$

$T_i^{\mathrm{b,e}}$ in (8) can be formulated as

$$T_i^{\mathrm{b,e}} = \frac{D_i}{\mu_i F^{\mathrm{m}}} \tag{11}$$

where $\mu_i \in [0, 1]$ denotes the fraction of the computation capability allocated to RU_i when offloading to the MEC server, and F^{m} (in cycles/s) denotes the computation capability of the MEC server.

E_i^{b} in (6) can be calculated as

$$E_i^{\mathrm{b}} = (1 - \lambda_i)E_i^0 + \lambda_i E_i^{\mathrm{b},0} \tag{12}$$

where $E_i^{\mathrm{b},0}$ denotes the energy consumption required when RU_i offloads its whole task to the MEC server. In this paper, we assume that the energy consumption of the RUs due to signal processing such as channel decoding is negligible [34], and examine the energy consumption of the RUs resulted from transmitting their task to the MEC server. Thus, we express $E_i^{\mathrm{b},0}$ as

$$E_i^{\mathrm{b},0} = \frac{p_i^{\mathrm{b}} I_i}{R_i}. \tag{13}$$

3) D2D Offloading Mode: When RU_i performs D2D offloading and transmits its task to SU_j, the task execution cost denoted by $U_{i,j}^{\mathrm{d}}$ is given by

$$U_{i,j}^{\mathrm{d}} = T_{i,j}^{\mathrm{d}} + \rho_i E_{i,j}^{\mathrm{d}} \tag{14}$$

where $T_{i,j}^{\mathrm{d}}$ and $E_{i,j}^{\mathrm{d}}$ denote, respectively, the task execution latency and energy consumption of RU_i when offloading to SU_j. $T_{i,j}^{\mathrm{d}}$ can be formulated as

$$T_{i,j}^{\mathrm{d}} = \max\left\{(1 - \lambda_{i,j})T_i^0, \lambda_{i,j} T_{i,j}^{\mathrm{d},0}\right\} \tag{15}$$

where $\lambda_{i,j} \in [0,1]$ denotes the task partition variable in the D2D offloading mode. More specifically, $\lambda_{i,j}$ is defined as the fraction of RU_i's task being executed at SU_j, and $1-\lambda_{i,j}$ denotes the fraction of RU_i's task being executed locally in the D2D offloading mode when RU_i is offloading to SU_j, $T_{i,j}^{\mathrm{d},0}$ denotes the task execution latency at SU_j when executing the whole task for RU_i, which can be expressed as

$$T_{i,j}^{\mathrm{d},0} = T_{i,j}^{\mathrm{d,t}} + T_{i,j}^{\mathrm{d,e}} \tag{16}$$

where $T_{i,j}^{\mathrm{d,t}}$ denotes the transmission latency required for RU_i to offload its whole task to SU_j and $T_{i,j}^{\mathrm{d,e}}$ denotes the task computation latency at SU_j. $T_{i,j}^{\mathrm{d,t}}$ is formulated as

$$T_{i,j}^{\mathrm{d,t}} = \frac{I_i}{R_{i,j}} \tag{17}$$

where $R_{i,j}$ denotes the achievable data rate of RU_i when transmitting to SU_j, which can be computed as

$$R_{i,j} = W^{\mathrm{d}} \log_2\left(1 + \frac{p_i^{\mathrm{d}} h_{i,j}}{\sigma^2}\right) \tag{18}$$

where p_i^{d} denotes the transmit power of RU_i when transmitting to SU_j and $h_{i,j}$ denotes the channel gain between RU_i and SU_j.

$T_{i,j}^{\mathrm{d,e}}$ in (16) can be formulated as

$$T_{i,j}^{\mathrm{d,e}} = \frac{D_i}{F_j^{\mathrm{d}}} \tag{19}$$

where F_j^{d} (in cycles/s) denotes the computation capability of SU_j.

$E_{i,j}^{\mathrm{d}}$ in (14) is given by

$$E_{i,j}^{\mathrm{d}} = (1-\lambda_{i,j}) E_i^0 + \lambda_{i,j} E_{i,j}^{\mathrm{d},0} \tag{20}$$

where $E_{i,j}^{\mathrm{d},0}$ denotes the energy consumption when RU_i offloads its whole task to SU_j, which can be calculated as

$$E_{i,j}^{\mathrm{d},0} = E_{i,j}^{\mathrm{d,t}} + E_{i,j}^{\mathrm{d,e}} \tag{21}$$

where $E_{i,j}^{\mathrm{d,t}}$ and $E_{i,j}^{\mathrm{d,e}}$ denote, respectively, the energy consumption occurred during data transmission and task execution when RU_i offloads its task to SU_j. $E_{i,j}^{\mathrm{d,t}}$ can be calculated as

$$E_{i,j}^{\mathrm{d,t}} = \frac{p_i^{\mathrm{d}} I_i}{R_{i,j}}. \tag{22}$$

$E_{i,j}^{\mathrm{d,e}}$ in (21) can be expressed as

$$E_{i,j}^{\mathrm{d,e}} = \delta_j D_i \left(F_j^{\mathrm{d}}\right)^2. \tag{23}$$

B. Optimization Constraints

To design the optimal joint computation offloading and resource allocation strategies, which minimizes the total task execution cost, we should consider a number of optimization constraints.

1) Computation Offloading Constraints: In this paper, we assume that each RU can only choose one of the three computation offloading modes, thus, we can express the computation offloading constraint as

$$\mathrm{C1}: x_i^0 + x_i + \sum_{j=1}^{M} x_{i,j} \le 1. \tag{24}$$

Constrained by transmission and computation resources, we further assume that one SU can only offer D2D offloading service to at most one RU, that is

$$\mathrm{C2}: \sum_{i=1}^{N} x_{i,j} \le 1. \tag{25}$$

It is apparent that the task partition variables λ_i and $\lambda_{i,j}$ are closely related to the computation offloading variables x_i and $x_{i,j}$, and the constraints on computation offloading and task partition can be expressed as

$$\mathrm{C3}: x_i \odot \lceil \lambda_i \rceil = 1 \tag{26}$$

$$\mathrm{C4}: \sum_{j=1}^{M} x_{i,j} \odot \lceil \lambda_{i,j} \rceil = 1 \tag{27}$$

where $\odot$ is the inclusive OR operation and $\lceil \cdot \rceil$ represents the ceiling function.

2) Resource Allocation Constraints: The fraction of the resources allocated to RU_i should meet the following constraints:

$$\mathrm{C5}: \hat{\eta}_i \in \left\{0, \frac{1}{L_0}, \frac{2}{L_0}, \ldots, 1\right\} \tag{28}$$

$$\mathrm{C6}: 0 \le \mu_i \le 1. \tag{29}$$

Due to the limited transmission resource at the BS and the computation resource at the MEC server, the sum of the resource allocation fractions should subject to an upper limit, i.e.,

$$\mathrm{C7}: \sum_{i=1}^{N} \hat{\eta}_i \le 1 \tag{30}$$

$$\mathrm{C8}: \sum_{i=1}^{N} \mu_i \le 1. \tag{31}$$

3) Task Execution Constraints: Stressing the latency sensitiveness, we assume that the task execution latency of RUs should meet a maximum allowable latency requirement, i.e.,

$$\mathrm{C9}: T_i \le T_i^{\max} \tag{32}$$

where T_i denotes the task execution latency of RU_i, which can be expressed as

$$T_i = x_i^0 T_i^0 + x_i T_i^{\mathrm{b}} + \sum_{j=1}^{M} x_{i,j} T_{i,j}^{\mathrm{d}}. \tag{33}$$

4) Transmission Rate Constraint: To guarantee efficient data transmission when RUs are performing task offloading, we assume that the transmission rate of individual RUs should meet a minimum data rate constraint, i.e.,

$$\mathrm{C10}: x_i R_i + \sum_{j=1}^{M} x_{i,j} R_{i,j} \ge R_i^{\min}, \text{ if } x_i^0 = 0 \tag{34}$$

where $R_i^{\min}$ denotes the minimum transmission rate of RU_i. In this paper, we assume that $R_i^{\min}$ is a given constant, which is determined by the characteristics of user tasks.

C. Optimization Problem

Considering the aforementioned objective function and optimization constraints, we formulate the task execution cost minimization-based joint computation offloading and resource

allocation problem as

$$\min_{x_i^0, x_i, x_{i,j}, \lambda_i, \lambda_{i,j}, \hat{\eta}_i, \mu_i} \quad U$$
$$\text{s.t.} \quad \text{C1} - \text{C10}. \tag{35}$$

Solving above optimization problem, we can obtain the joint computation offloading and resource allocation strategies.

V. Solution of the Optimization Problem: Single RU Case

In this section, we consider a simple case that only one RU needs to execute its computation task, i.e., $N = 1$. As no resource contention occurs in this case, and all the bandwidth resource of the BS and the computation capability of the MEC server can be allocated to the RU, i.e., $\hat{\eta_1} = 1$ and $\mu_1 = 1$, the original joint computation offloading and resource allocation problem reduces to the computation offloading problem of one RU, which can be expressed as

$$\min_{x_1^0, x_1, x_{1,j}, \lambda_1, \lambda_{1,j}} \quad U_1$$
$$\text{s.t.} \quad \text{C1–C4, C9, C10 in (35)}. \tag{36}$$

According to (2), U_1 consists of three terms corresponding to different computation offloading modes, i.e., $U_1 = x_1^0 U_1^0 + x_1 U_1^{\mathrm{b}} + \sum_{j=1}^{M} x_{1,j} U_{1,j}^{\mathrm{d}}$. Although U_1^0 can be calculated easily, U_1^{b} and $U_{1,j}^{\mathrm{d}}$ are the functions of λ_1 and $\lambda_{1,j}$ respectively, which cannot be calculated exactly without determining λ_1 and $\lambda_{1,j}$. To tackle this problem, we first consider the MEC offloading mode and D2D offloading mode individually and design the optimal task partition strategy, i.e., λ_1^* and $\lambda_{1,j}^*$ so as to minimize the task execution cost of the particular computation offloading mode.

For the MEC offloading mode, the task execution cost minimization problem of RU_1 is given by

$$\min_{\lambda_1} \quad U_1^{\mathrm{b}}$$
$$\text{s.t.} \quad 0 < \lambda_1 \leq 1. \tag{37}$$

The above-formulated problem is a single variable optimization problem, which can be solved by various mathematical methods. In this paper, applying an extensive search method, we examine the function U_1^{b} over different $\lambda_1 \in [0, 1]$ to obtain the optimal λ_1^* and the corresponding task execution cost, denoted by $U_1^{\mathrm{b},*}$.

Similarly, for the D2D offloading mode, we assume that RU_1 offloads to one particular SU, e.g., SU_j, $1 \leq j \leq M$, and formulate the task execution cost minimization problem of RU_1, i.e.,

$$\min_{\lambda_{1,j}} \quad U_{1,j}^{\mathrm{d}}$$
$$\text{s.t.} \quad 0 < \lambda_{1,j} \leq 1 \tag{38}$$

and solve the problem by means of the extensive search method to obtain the optimal $\lambda_{1,j}^*$ and the corresponding task execution cost, denoted by $U_{1,j}^{\mathrm{d},*}$.

Given $U_1^0, U_1^{\mathrm{b},*}$, and $U_{1,j}^{\mathrm{d},*}$, the optimal computation offloading strategy can be determined by selecting the one offering the minimum task execution cost, i.e.,

$$x_1^{0,*} = \begin{cases} 1, \text{ if } U_1^0 \leq U_1^{\mathrm{b},*} \text{ and } U_1^0 \leq U_{1,j}^{\mathrm{d},*} \quad \forall j \\ 0, \text{ others} \end{cases} \tag{39}$$

$$x_1^* = \begin{cases} 1, \text{ if } U_1^{\mathrm{b},*} < U_1^0 \text{ and } U_1^{\mathrm{b},*} \leq U_{1,j}^{\mathrm{d},*} \quad \forall j \\ 0, \text{ others} \end{cases} \tag{40}$$

$$x_{1,j}^* = \begin{cases} 1, \text{ if } U_{1,j}^{\mathrm{d},*} < U_1^0 \text{ and } U_{1,j}^{\mathrm{d},*} < U_1^{\mathrm{b},*} \quad \forall j \\ 0, \text{ others.} \end{cases} \tag{41}$$

VI. Solution of the Optimization Problem: Multiple RUs Case

In this section, we extend the single RU case to a more general multiple RUs case, i.e., $N > 1$, and solve the optimization problem formulated in (35), as the optimization problem is a mixed integer nonlinear optimization problem, which cannot be solved conveniently. In this paper, we propose a heuristic algorithm, which successively solves two subproblems, i.e., computation offloading subproblem and resource allocation subproblem, and obtain the joint computation offloading and resource allocation strategies.

A. Computation Offloading Subproblem

In this section, we assume that no resource sharing occurs at the BS and the MEC server, i.e., $\hat{\eta_i} = 1$ and $\mu_i = 1, 1 \leq i \leq N$. Hence, the original joint computation offloading and resource allocation problem reduces to the computation offloading subproblem, which can be formulated as

$$\min_{x_i, x_{i,j}, \lambda_i, \lambda_{i,j}} \quad U$$
$$\text{s.t.} \quad \text{C1–C4, C9, C10 in (35)}. \tag{42}$$

To solve the optimization problem, we further split the problem and successively conduct the following two procedures, i.e., task partition and computation offloading mode selection. More specifically, we first set $x_i = 1$ and $x_{i,j} = 1$, and design the optimal task partition strategies, i.e., λ_i^* and $\lambda_{i,j}^*$ for $\mathrm{RU}_i, 1 \leq i \leq N$, following the similar manner as discussed in Section V. Then, given λ_i^* and $\lambda_{i,j}^*$, we optimally select the computation offloading modes for RUs through conducting the optimal matching between the RUs and the offloading modes.

1) Optimal Task Partition Strategies: In this section, we examine the task execution cost of RUs in various computation offloading modes.

According to the definition of task execution cost specified in (3), (6), and (14), we can see that while the task execution cost for complete local computing, i.e., U_i^0, can be calculated explicitly for RU_i, the task execution cost for conducting both MEC and D2D offloading, i.e., U_i^{b} and $U_{i,j}^{\mathrm{d}}$, is a function of task partition variables, i.e., λ_i and $\lambda_{i,j}$, which cannot be calculated directly. However, it can be shown that the optimal task partition strategy of one RU at a certain computation offloading mode is independent on the characteristics of other offloading modes and other RUs, thus we will be able to examine the task execution cost of each RU at particular offloading modes, and determine the optimal task partition strategy accordingly. More specifically, as U_i^{b} is a single-variable function of λ_i, we can apply extensive search method to obtain λ_i^* and the corresponding minimum task execution cost $U_i^{\mathrm{b},*}$, $1 \leq i \leq N$. Similarly, for the D2D offloading mode, we can calculate $\lambda_{i,j}^*$, and obtain the corresponding minimum task execution cost $U_{i,j}^{\mathrm{d},*}$, $1 \leq i \leq N$, $1 \leq j \leq M$.

2) Computation Offloading Mode Selection: It can be understood in a straightforward manner that in the case that one RU achieves the minimum task execution cost when performing complete local computing compared with computation offloading modes, the RU may tend to execute its task locally. Thus, we can first determine the complete local computing mode for RUs simply by comparing the task execution cost in a complete local computing mode with those in computation offloading modes.

As the task execution cost of RUs in complete local computing mode can be calculated based on (3), and the task execution cost of RUs in various computation offloading modes can be calculated according to the obtained optimal task partition strategies, we can then compare the task execution cost of RUs in different computing modes. In the case that RU_i achieves the minimum task execution cost when conducting complete local computing, i.e., $U_i^0 \leq U_i^{\mathrm{b}}$ and $U_i^0 \leq U_{i,j}^{\mathrm{d}}, \forall 1 \leq j \leq M$, we will assign the complete local computing mode to RU_i, i.e., $x_i^{0,*} = 1$, $x_i^* = 0$, $\lambda_i^* = 0$, $x_{i,j}^* = 0$, and $\lambda_{i,j}^* = 0, 1 \leq i \leq N, 1 \leq j \leq M$.

In the case that RU_i achieves the minimum task execution cost when offloading to $\mathrm{SU}_{j'}$, i.e., $U_{i,j'}^{\mathrm{d}} \leq U_i^0$, $U_{i,j'}^{\mathrm{d}} \leq U_i^{\mathrm{b}}$, and $U_{i,j'}^{\mathrm{d}} \leq U_{i,j}^{\mathrm{d}}, \forall 1 \leq j \leq M$, $j \neq j'$, and no other RUs achieve the minimum cost when offloading to $\mathrm{SU}_{j'}$, we will assign the D2D offloading mode to RU_i, and set $\mathrm{SU}_{j'}$ as the D2D offloading peer of RU_i, i.e., $x_{i,j'}^* = 1$, $x_i^{0,*} = 0$, $x_i^* = 0$, $\lambda_i^* = 0$, $x_{i,j}^* = 0$, $1 \leq i \leq N, 1 \leq j \neq j' \leq M$.

3) Kuhn–Munkres Algorithm-Based Optimal Computation Offloading Strategies: Removing the RUs, which have been assigned the complete local computing mode or D2D offloading mode from the RU set, we denote the set of the remaining RUs as Φ_{RU} and the number of the remaining UE as N'. Similarly, we denote the set of SUs, which have not been selected as the D2D offloading peer for the RUs, as Φ_{SU}, we further study the computation offloading strategy of the RUs in Φ_{RU}.

We rewrite the total task execution cost of the remaining RUs as follows:

$$U^* = \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}} U_i^* \tag{43}$$

where U_i^* denotes the optimal task execution cost of $\mathrm{RU}_i \in \Phi_{\mathrm{RU}}$ in terms of λ_i and $\lambda_{i,j}$, which is given by

$$U_i^* = x_i^0 U_i^0 + x_i U_i^{\mathrm{b},*} + \sum_{\mathrm{SU}_j \in \Phi_{\mathrm{SU}}} x_{i,j} U_{i,j}^{\mathrm{d},*}. \tag{44}$$

The computation offloading subproblem of the remaining UE can be formulated as

$$\begin{aligned} \min_{x_i^0, x_i, x_{i,j}} \quad & U^* \\ \text{s.t.} \quad & x_i^0 + x_i + \sum_{\mathrm{SU}_j \in \Phi_{\mathrm{SU}}} x_{i,j} \leq 1 \\ & \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}} x_{i,j} \leq 1, \ \mathrm{SU}_j \in \Phi_{\mathrm{SU}}. \end{aligned} \tag{45}$$

The above-formulated optimization problem can be regarded as a matching problem between RUs and offloading modes, however, as multiple RUs may choose to offload to the MEC server, the problem involves both one-to-one matching and one-to-many matching, which cannot be solved using typical matching algorithms. To this end, we apply network virtualization scheme and virtualize the physical MEC server into N' virtual MEC servers. Assuming each virtual MEC server is allowed to perform computation offloading for at most one RU, and defining the binary variable $x_{i,k}^{\mathrm{v}}$ to denote the computation offloading variable when RU_i offloads its task to the kth virtual MEC server, we can rewrite U_i^* as

$$U_i^* = x_i^0 U_i^0 + \sum_{k=1}^{N'} x_{i,k}^{\mathrm{v}} U_{i,k}^{\mathrm{b},*} + \sum_{\mathrm{SU}_j \in \Phi_{\mathrm{SU}}} x_{i,j} U_{i,j}^{\mathrm{d},*} \tag{46}$$

where $U_{i,k}^{\mathrm{b},*} = U_i^{\mathrm{b},*}, 1 \leq k \leq N'$. The optimization problem formulated in (45) can then be rewritten as

$$\begin{aligned} \min_{x_i^0, x_{i,k}^{\mathrm{v}}, x_{i,j}} \quad & U^* \\ \text{s.t.} \quad & x_i^0 + \sum_{k=1}^{N'} x_{i,k}^{\mathrm{v}} + \sum_{\mathrm{SU}_j \in \Phi_{\mathrm{SU}}} x_{i,j} \leq 1 \\ & \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}} x_{i,j} \leq 1 \quad \forall \mathrm{SU}_j \in \Phi_{\mathrm{SU}}. \end{aligned} \tag{47}$$

The optimization problem formulated in (47) is now in the form of one-to-one matching problem in a bipartite graph, which can be solved by Kuhn–Munkres algorithm [35].

To solve the problem, we first map the optimization problem into a weighted complete bipartite graph with bipartite division $G^0 = (V_1, V_2, E, W)$, where V_1 and V_2 are the set of vertices, V_1 denotes the remaining RUs, which have not be assigned an offloading mode, and V_2 represents various computation offloading modes, $E = \{e(v_1, v_2)\}$ denotes the set of edges, which connect one vertex in V_1 to another vertex in V_2, and $W = \{w(v_1, v_2)\}$ denotes the weight of the edge $e(v_1, v_2) \in E$, which is the minimum task execution cost given in different computation and offloading mode.

The steps for solving the computation offloading subproblem based on the Kuhn–Munkres algorithm are as follows:

1) find an initial feasible vertex labeling $l(v_1)$;
2) for given $l(v_1)$, determine G_l^0 from G^0 and select a maximum matching H;
3) if H is an optimal matching, then the optimization problem formulated in (47) is solved. Otherwise, a label having not being allocated by H is selected in G_l^0. Set $P = V_1$, and $T = \Psi$, which denotes the empty set;
4) $N_{G_l^0}(S)$ denotes the collection of points, which connect with S in G_l^0. If $N_{G_l^0}(P) \neq T$, go to step 3. Otherwise, $N_{G_l^0}(P) = T$. Find $\triangle = \min_{v_1, v_2}(l(v_1) | l(v_1) + l(v_2) - w(v_1, v_2), v_1 \in P, v_2 \in V_2 - T)$ and define

$$l'(v_1) = \begin{cases} l(v_1) - \Delta, & v_1 \in P \\ l(v_1) + \Delta, & v_1 \in T \\ l(v_1), & \text{others.} \end{cases} \tag{48}$$

5) replace $l(v_1)$ by $l'(v_1)$, go back to step 2.

Conducting above process iteratively, we can obtain the optimal computation offloading strategies, i.e., $x_i^{0,*}, x_{i,k}^{\mathrm{v},*}, x_{i,j}^*$. For a given set of computation offloading strategies, we then solve the resource allocation subproblem for the RUs selecting the MEC offloading mode simultaneously.

B. Resource Allocation Subproblem

Let $\Phi_{\mathrm{RU}}^{\mathrm{b}} = \{\mathrm{RU}_i | x_{i,k}^{\mathrm{v},*} = 1\}$ denote the set of RUs choosing to offload their tasks to the MEC server. In the case that more than one RU selects the MEC offloading mode, i.e., $\sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} \sum_{k=1}^{N'} x_{i,k}^{\mathrm{v},*} > 1$, resource sharing occurs at the MEC server and the ORA strategy should be designed.

In this section, we formulate the resource allocation subproblem and solve the subproblem by means of the Lagrange dual method to obtain the optimal fraction of the bandwidth and computation resource. The resource allocation subproblem can be formulated as

$$\min_{\hat{\eta}_i, \mu_i} \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} U_i^{\mathrm{b}} \quad \text{s.t.} \quad \text{C5–C10 in (35)}. \tag{49}$$

Applying the relaxation method, we first transform the discrete optimization variable $\hat{\eta}_i \in \{0, \frac{1}{L_0}, \frac{2}{L_0}, \ldots, 1\}$ in (35) into a continuous variable, i.e., $0 \le \tilde{\eta}_i \le 1$. The original optimization problem can then be rewritten as

$$\begin{aligned} \min_{\tilde{\eta}_i, \mu_i} \quad & \sum_{\mathrm{RU}_i \in \Phi_{RU}^{\mathrm{b}}} U_i^{\mathrm{b}} \\ \text{s.t.} \quad & \mathrm{C6}, \mathrm{C8–C10 \text{ in } (35)} \\ & \mathrm{C11}: 0 \le \tilde{\eta}_i \le 1 \\ & \mathrm{C12}: \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} \tilde{\eta}_i \le 1. \end{aligned} \tag{50}$$

It can be proved that the optimization problem formulated in (50) is a convex problem, which can be solved by using the Lagrange dual method [36]. The corresponding Lagrange function can be obtained as

$$\begin{aligned} L(\tilde{\eta}_i, \mu_i, \alpha_i, \beta_i, \gamma_i, \theta_i, \varepsilon, \tau) = & \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} (T_i^{\mathrm{b}} + \rho_i E_i^{\mathrm{b}}) \\ & + \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} \alpha_i \left(T_i^{\mathrm{b}} - T_i^{\max}\right) + \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} \beta_i \left(R_i^{\min} - R_i\right) \\ & + \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} \gamma_i (\tilde{\eta}_i - 1) + \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} \theta_i (\mu_i - 1) \\ & + \varepsilon \left(\sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} \tilde{\eta}_i - 1 \right) + \tau \left(\sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} \mu_i - 1 \right) \end{aligned} \tag{51}$$

where $\alpha_i, \beta_i, \gamma_i, \theta, \varepsilon, \tau$ are non-negative Lagrange multipliers. Thus, the Lagrange dual problem can be formulated as

$$\max_{\alpha_i, \beta_i, \gamma_i, \theta_i, \varepsilon, \tau} \min_{\tilde{\eta}_i, \mu_i} L(\tilde{\eta}_i, \mu_i, \alpha_i, \beta_i, \gamma_i, \theta_i, \varepsilon, \tau) \quad \text{s.t.} \quad \alpha_i, \beta_i, \gamma_i, \theta_i, \varepsilon, \tau > 0. \tag{52}$$

For a given set of Lagrange multipliers, the optimal fraction of bandwidth resource and computation capability can be obtained as

$$\tilde{\eta}_i^* = \left[\sqrt{\frac{\left(1 + \alpha_i + \rho_i p_i^{\mathrm{b}}\right) I_i}{\left(\gamma_i + \varepsilon - \beta_i\right) g_i^{\mathrm{b}}}} \right]^+ \tag{53}$$

$$\mu_i^* = \left[\sqrt{\frac{(1 + \alpha_i) D_i}{(\theta_i + \tau) F^{\mathrm{m}}}} \right]^+ \tag{54}$$

where $g_i^{\mathrm{b}} = W^{\mathrm{b}} \log_2 (1 + \frac{p_i^{\mathrm{b}} h_i}{\sigma^2})$ and $[x]^+ = \max\{x, 0\}$.

The Lagrange multipliers can be updated based on the gradient method [36], i.e.,

$$\alpha_i(t+1) = \left[\alpha_i(t) - \pi_1(t)\left(T_i^{\mathrm{b}} - T_i^{\max}\right)\right]^+ \tag{55}$$

$$\beta_i(t+1) = \left[\beta_i(t) - \pi_2(t)\left(R_i^{\min} - R_i\right)\right]^+ \tag{56}$$

$$\gamma_i(t+1) = \left[\gamma_i(t) - \pi_3(t)\left(\tilde{\eta}_i - 1\right)\right]^+ \tag{57}$$

$$\theta_i(t+1) = \left[\theta_i(t) - \pi_4(t)\left(\mu_i - 1\right)\right]^+ \tag{58}$$

$$\varepsilon(t+1) = \left[\varepsilon(t) - \pi_5(t)\left(\sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} \tilde{\eta}_i - 1\right)\right]^+ \tag{59}$$

$$\tau(t+1) = \left[\tau(t) - \pi_6(t)\left(\sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} \mu_i - 1\right)\right]^+ \tag{60}$$

where t denotes the iteration index and $\pi_q(t)$ are stepsizes, $1 \le q \le 6$. The convergence condition of the algorithm is given by

$$\begin{aligned} \sum_{\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}} & [|\alpha_i(t+1) - \alpha_i(t)| + |\beta_i(t+1) - \beta_i(t)| \\ & + |\gamma_i(t+1) - \gamma_i(t)| + |\theta_i(t+1) - \theta_i(t)|] \\ & + |\varepsilon(t+1) - \varepsilon(t)| + |\tau(t+1) - \tau(t)| \le \varpi \end{aligned} \tag{61}$$

where ϖ denotes the maximum tolerance. After obtaining the optimal fraction of bandwidth resource, we recover the relaxed variable $\tilde{\eta}_i^*$ to original variable $\hat{\eta}_i^*$ as

$$\hat{\eta}_i^* = \frac{\left[\tilde{\eta}_i^* L_0 + \frac{1}{2L_0}\right]}{L_0} \tag{62}$$

where $[\cdot]$ represents the round function. The Lagrange dual method based resource allocation algorithm is summarized in Algorithm 1.

Replacing $\hat{\eta}_i$ and μ_i by $\hat{\eta}_i^*$ and μ_i^*, respectively, in U_i^{b}, we will be able to obtain the optimal task execution cost of RU_i at the MEC offloading mode when sharing bandwidth and computation resource with other RUs. As it is possible that the performance of task execution of one RU at the MEC offloading mode becomes worse than that obtained at the complete local computing mode due to resource sharing, in this case, we should reassign the complete local computing mode to those RUs.

C. Complexity Analysis

In this section, we analyze the computation complexity of the proposed algorithm. As two subproblems, i.e., computation offloading subproblem and resource allocation subproblem are successively solved, we examine the complexity of solving the two subproblems, respectively.

For the computation offloading subproblem, under the assumption that the continuous task partition variables λ_i and $\lambda_{i,j}$ are uniformly discretized into Q values, the number of operations for all the RUs to minimize execution cost at various offloading modes using extensive search is $N(M+1)Q$. The computational complexity is $O(N(M+1)Q)$. As we virtualize

Algorithm 1: Lagrange Dual Method Based Resource Allocation Algorithm.

Set the maximum number of iterations $t_{\max}$ and maximum tolerance ϖ
Initialize Lagrange multipliers α_i, β_i, γ_i, θ_i, ε and τ
Set $t = 0$
repeat
  for $\mathrm{RU}_i \in \Phi_{\mathrm{RU}}^{\mathrm{b}}$ **do**
    Calculate $\tilde{\eta}_i$ according to (53);
    Calculate μ_i according to (54);
    Update α_i, β_i, γ_i, θ_i, ε and τ according to (55)−(60);
    if (61) achieves
      then
        Convergence = **true**
        return $\tilde{\eta}_i^* = \tilde{\eta}_i$, $\mu_i^* = \mu_i$
      else
        $t = t + 1$
      end if
  end for
until Convergence = **true** or $t = t_{\max}$
Recover $\tilde{\eta}_i^*$ to $\hat{\eta}_i^*$ according to (62)

the physical BS into N' virtual BSs, the complexity of the Kuhn–Munkres algorithm is $O(Z^3)$ with $(11Z^3 + 12Z^2 + 31Z)/6$ operations, where $Z = N' + M + 1$ [35].

To solve the formulated resource allocation subproblem, at each iteration, we need to perform $|\Phi_{\mathrm{RU}}^{\mathrm{b}}|$ operations, where $|\cdot|$ denotes the cardinality of one set. As each RU updates Lagrange multipliers according to (55)–(60), the number of operations for updating the Lagrange multiplier is $4|\Phi_{\mathrm{RU}}^{\mathrm{b}}| + 2$. Assume K iterations are required for the algorithm to achieve convergence, the number of operations for conducting resource allocation is $4K|\Phi_{\mathrm{RU}}^{\mathrm{b}}| + 2K$. The corresponding computational complexity is $O(|\Phi_{\mathrm{RU}}^{\mathrm{b}}|K)$.

Accordingly, the overall complexity for conducting both computation offloading and resource allocation is $O(Z^3)$.

VII. Simulation Results

In this section, we examine the performance of the proposed joint computation offloading and resource allocation algorithm, and compare our proposed algorithm with previously proposed schemes. In the simulation, we consider a cellular D2D MEC system consisting of one BS and a number of users. The size of the simulation region is 1000 m×1000 m, and the users are uniformly located in the simulation area. Other parameters used in the simulation are summarized in Table I. To characterize various features and requirements of the RUs, SUs, and the computation tasks in the simulation, we randomly select the input data size, the required number of CPU cycles, the computation capability of the RUs, SUs, and the MEC server from a set consisting of two elements, as shown in Table I. The simulation results are averaged over 1000 independent experiments.

In Fig. 3(a), the task execution cost versus the input data size of the task is evaluated for our proposed algorithm, the algorithm proposed in [5] and the optimal scheme. To obtain the optimal solution, we first calculate the task execution cost of RUs at different computing modes. By comparing the cost at various computing mode, the RUs, which achieve the minimum task execution cost under local computing are assigned a complete local computing mode. We then enumerate all possible MEC offloading strategies for the RUs. In the case that more than one RU chooses to offload its task to the MEC server, we apply Lagrange dual method to calculate the ORA strategies. Although for the RUs choosing to conduct D2D offloading, we then determine the optimal D2D offloading peer by means of the Kuhn–Munkres algorithm. In the simulation, we set the number of RUs and SUs both as 5. It can be seen from the figure that the task execution cost increases with the increase of the input data size for both algorithms. This is because larger input data size requires longer task execution latency and larger energy consumption for completing task computation, thus resulting in larger task execution cost. Comparing the results obtained from our proposed scheme and the one proposed in [5], we can see that our proposed scheme offers better performance. The reason is that the scheme proposed in [5] aims to achieve the minimum execution latency, thus may require higher energy consumption,

TABLE I
Simulation Parameters

Parameters	Value
Bandwidth of BS (W^{b})	10 MHz
Number of subchannels (L_0)	5∼35
Bandwidth of D2D link (W^{d})	5 MHz
Distance of D2D transmission	300 m
Transmit power of RU_i on cellular link (p_i^{b})	600 mW
Transmit power of RU_i on D2D link (p_i^{d})	200 mW
Path loss factor	3
Noise power (σ^2)	-75 dBm
Computation capability of MEC server (F^{m})	30 Gigacycles/s
Scalar weight (ρ_i)	0.01 s/J
Input data size (I_i)	$\{1, 2\}$ Mbits
Required number of CPU cycles (D_i)	$\{5, 6\}$ Gigacycles
Computation capability of RU_i (F_i)	$\{0.8, 1\}$ Gigacycles/s
Computation capability of SU_j (F_j^{d})	$\{1.2, 1.5\}$ Gigacycles/s

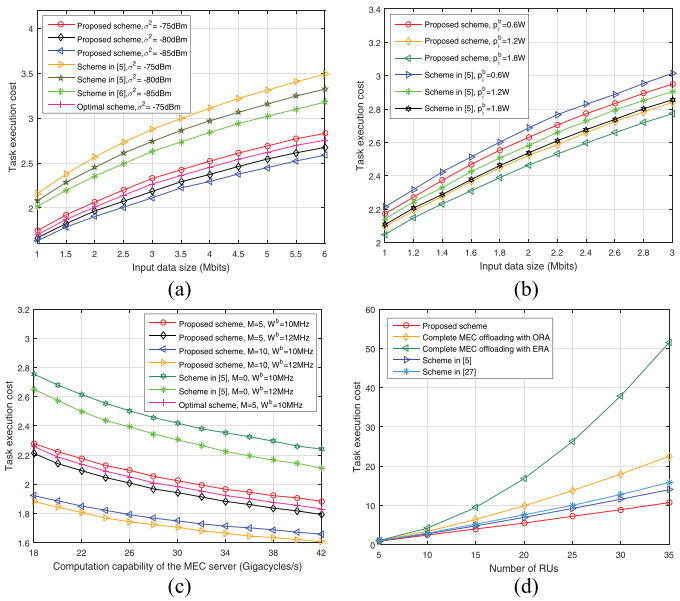

Fig. 3. (a) Task execution cost versus input data size (different noise power). (b) Task execution cost versus input data size (different transmit power). (c) Task execution cost versus computation capability of the MEC server. (d) Task execution cost versus number of RUs.

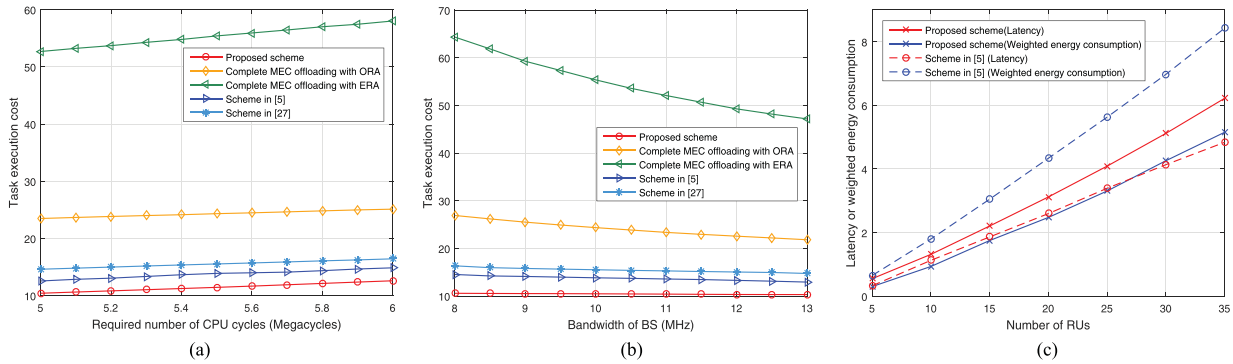

Fig. 4. (a) Task execution cost versus required number of CPU cycles. (b) Task execution cost versus bandwidth of BS. (c) Latency or weighted energy consumption versus number of RUs.

causing larger task execution cost in turn. We can also see that there is only a small gap between our proposed scheme and the optimal scheme.

Fig. 3(b) shows the task execution cost versus the input data size for different transmit power of RUs on a cellular link. The number of RUs is chosen as 5 in examining the results. For comparison, we plot the task execution cost obtained from our proposed scheme and the scheme proposed in [5]. It can be observed from the figure that, for both schemes, the task execution cost increases with the increase of input data size. We can also see from the figure that the task execution cost decreases with the increase of transmit power on cellular link. This is because the higher transmit power offers better transmission performance between the RUs and the BS, thus resulting in smaller task transmission latency. Although the energy consumption increases with the increase of transmit power, however, it can be shown that for relatively small transmit power as the case for mobile applications, the increase of energy consumption is relatively small. Therefore, smaller task execution cost can be obtained for slightly high transmit power as can be observed from the figure.

Fig. 3(c) shows the task execution cost versus the computation capacity of the MEC server obtained from different number of SUs and different bandwidth resource of the BS. To plot the figure, the number of RUs is chosen as 5. From the figure, we can see that the task execution cost decreases with the increase of the computation capacity of the MEC server for the proposed scheme and scheme proposed in [5]. This is because the higher computation capacity of the MEC server results in better task execution performance and lower task execution cost in turn. Comparing the results obtained from our proposed scheme, the scheme proposed in [5] and the optimal scheme, we can see that our proposed scheme outperforms the previously proposed scheme and the gap between our proposed scheme and the optimal scheme is quite small. It can also be seen from the figure that for larger number of SUs and larger bandwidth of the BS, lower task execution cost can be obtained, which is benefited from better task transmission and computation offloading performance.

In Fig. 3(d), we examine the task execution cost versus the number of RUs obtained from our proposed scheme, the scheme proposed in [5] and [27] and two benchmark schemes, i.e., complete MEC offloading with ORA and complete MEC offloading with equal resource allocation (ERA). For both ORA and ERA schemes, we assume that all the RUs offload their whole tasks to the MEC server. In the ORA scheme, the computation resource of the MEC server is allocated in an optimal manner, whereas in the ERA scheme, the computation resource of the MEC server is divided equally and assigned to the RUs. It can be seen from the figure that the task execution cost increases with the increase of the number of RUs. Comparing various schemes, we can see that the proposed scheme offers the lowest task execution cost, and the performance gap between the proposed scheme and other four schemes increases as the number of RUs increases. This is mainly due to the strong resource competition at the MEC server when the number of RUs becomes large.

Fig. 4(a) shows the task execution cost versus the required number of CPU cycles obtained from our proposed scheme, the schemes proposed in [5] and [27] and two benchmark schemes. The number of RUs is set as 40 in the simulation. We can see from the figure that for various schemes, the task execution cost increases with the increase of required computation resource. The reason is that larger required computation resource results in longer task execution latency and larger energy consumption for task computation. Comparing the results obtained from the proposed scheme and other schemes, we can see that the proposed scheme offers the lowest task execution cost than other schemes, indicating that the proposed joint computation offloading and resource allocation results in desired performance enhancement. It can also be observed from the figure that the task execution cost of the ORA scheme is smaller than that of the ERA scheme, this is benefited from the efficient resource utilization at the BS and the MEC server.

In Fig. 4(b), we examine the task execution cost versus the available bandwidth of BS. The number of RUs is set as 40 in the simulation. It can be seen from the figure that the task execution cost decreases with the increase of available bandwidth of the BS. Comparing the results obtained from the proposed scheme and four other schemes, we can see that the proposed scheme outperforms the other schemes.

Fig. 4(c) shows the task execution latency and the energy consumption cost versus the number of RUs obtained from our proposed scheme and the scheme proposed in [5]. We can see from the figure that the scheme proposed in [5] offers slightly better latency performance however, requires much higher energy consumption compared with our proposed scheme. This is because the scheme proposed in [5] aims to minimize the task execution latency, and the latency and energy consumption are both taken into consideration in our proposed scheme, thus is capable of achieving a tradeoff between latency and energy consumption.

VIII. Conclusion

In this paper, we study the joint computation offloading and resource allocation in a cellular D2D MEC system. To achieve efficient information interaction and task management, we propose a joint task management architecture and formulate the joint computation offloading and resource allocation as an optimization problem, which minimizes the task execution cost of all the RUs. By solving the formulated optimization problem, the joint computation offloading and resource allocation strategies are obtained. We also examine the performance of our proposed scheme and compare it with previously proposed schemes and the optimal one. Simulation results show that our proposed scheme outperforms previously proposed schemes and the gap between the proposed scheme and the optimal one is relatively small, thus demonstrating the effectiveness of the proposed scheme.

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, Fourth Quarter 2017.

[2] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching, and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

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

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," *ETSI White Paper*, vol. 11, pp. 3–4, 2015.

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

[6] 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.

[7] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[8] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory*, 2016, pp. 1451–1455.

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

[10] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5G," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6398–6409, Jul. 2018.

[11] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 726–738, Sep.-Oct. 1 2019.

[12] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

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

[14] S. Yu, X. Wang, and R. Langar, "Computation offloading for mobile edge computing: A deep learning approach," in *Proc. IEEE Int. Symp. Pers., Indoor Mobile Radio Commun.*, 2017, pp. 1–6.

[15] S. Bi and Y. J. A. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

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

[17] M. Li, S. Yang, Z. Zhang, J. Ren, and G. Yu, "Joint subcarrier and power allocation for OFDMA based mobile edge computing system," in *Proc. IEEE Int. Symp. Pers., Indoor Mobile Radio Commun.*, 2017, pp. 1–6.

[18] J. Guo, Z. Song, Y. Cui, Z. Liu, and Y. Ji, "Energy-efficient resource allocation for multi-user mobile edge computing," in *Proc. IEEE Global Telecommun. Conf.*, 2017, pp. 1–7.

[19] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Global Telecommun. Conf.*, 2016, pp. 1–6.

[20] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and CPU time allocation for mobile edge computing," in *Proc. IEEE Global Telecommun. Conf.*, 2016, pp. 1–6.

[21] X. Zhang, Y. Mao, J. Zhang, and K. B. Letaief, "Multi-objective resource allocation for mobile edge computing systems," in *Proc. IEEE Int. Symp. Pers., Indoor Mobile Radio Commun.*, 2017, pp. 1–5.

[22] Z. Zhu *et al.*, "Fair resource allocation for system throughput maximization in mobile edge computing," *IEEE Access*, vol. 6, pp. 5332–5340, 2018.

[23] 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 Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[24] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

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

[26] T. T. Nguyen and B. L. Long, "Joint computation offloading and resource allocation in cloud based wireless hetnets," in *Proc. IEEE Global Telecommun. Conf.*, 2017, pp. 1–6.

[27] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19324–19337, 2018.

[28] M. H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.

[29] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[30] X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21355–21367, 2017.

[31] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752–764, Jan. 2018.

[32] 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 Trans. Commun.*, vol. 66, no. 4, pp. 1594–1608, Apr. 2018.

[33] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[34] A. Yadav, T. M. Nguyen, and W. Ajib, "Optimal energy management in hybrid energy small cell access points," *IEEE Trans. Commun.*, vol. 64, no. 12, pp. 5334–5348, Dec. 2016.

[35] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, Mar. 1957.

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