IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 2, 2019, accepted July 9, 2019, date of publication July 12, 2019, date of current version August 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2928377

Multi-User and Multi-Task Offloading Decision Algorithms Based on Imbalanced Edge Cloud

WEN-JIANG FENG, CHONG-HAI YANG, AND XIAO-SHAN ZHOU
College of Microelectronics and Communication Engineering, Chongqing University, Chongqing 400044, China

Corresponding author: Chong-Hai Yang (yangch@cqu.edu.cn)

This work was supported in part by the Chongqing Basic Science and Frontier Technology Research Project under Grant cstc2017jcyjBX0047.

ABSTRACT The mobile edge computing (MEC) technology can provide mobile users (MU) with high reliability and low time-delay computing and communication services. The imbalanced edge cloud deployment can better adapt to the non-uniform spatial-time distribution of tasks and reduce the deployment cost of edge cloud servers. For multi-user and multi-task offloading decision based on the imbalanced edge cloud, a new offloading cost criteria, based on the tradeoff among time delay-energy consumption-cost, is designed to quantify the user experience of task offloading and to be the optimization target of offloading decision. Both the optimization problems of minimizing the sum offloading costs for all MUs (efficiency-based) and minimizing the maximal offloading cost per MU (fairness-based) are discussed. Efficiency-based offloading decision algorithm [centralized greedy algorithm (CGA) and modified greedy algorithm (MGA)] and fairness-based offloading decision algorithm [fairness-based greedy algorithm (FGA)] are proposed, respectively, and the performance bounds of the algorithm are analyzed. The simulation results show that the offloading cost of the MGA is lower than the CGA, the efficiency of resource utilization of the CGA is higher than that of the FGA, and the fairness of the FGA is stronger than that of the CGA.

INDEX TERMS Mobile edge computing, edge cloud deployment, offloading decision, greedy algorithm.

I. INTRODUCTION

Mobile Edge Computing (MEC) technology provides mobile users (MU) with high reliability, low time-delay computing and communication services by deploying Edge Cloud Server (ECS) at the edge of the mobile network [1]. It is designed to solve contradictions between the latency sensitive services and the limited terminal processing capability, the intensive computations and the weakness of bearing capacity, the abundant cloud computing resources and the restricted access capability, the significantly growing number of mobile broadband service and the pipelined bearer network. MEC promotes the development of real-time and highly reliable services in the field of industrial Internet of Things, Intelligent Connected and Internet of Vehicles. But, due to multi-user and multi-task competition for communication and computing resources, user experience relies on reasonable offloading decisions.

In terms of MEC offloading decisions, the centralized and distributed offloading decision algorithm is studied, where the weighted sum of delay-energy is deployed to characterize the offloading costs in [2]. For single-user MEC system, the energy consumption of offloading computing and local computing determines whether to offload in [3]. In the multi-user MEC system, the offloading decision is related to the offloading user set, and the coupling relationship between the multi-user offloading decision and the resource configuration is derived in [4]. In the case of the MEC architecture, each access point (AP) or base station (BS) is equipped with a separate ECS in [5]–[7], while multiple APs sharing access to the same ECS in [8], [9]. The system models of the above studies all use balanced deployment of edge cloud servers. In practical applications, using unbalanced edge cloud deployment is a better choice, due to the non-uniform distribution of tasks and the cost of ECS, that is, multiple APs share multiple ECSs in [10]. But there will be a series of new problems: Firstly, due to the mutual coupling of shared ECS and AP, the offloading decision is more complicated. Secondly, the optimization goal has an important impact on the offloading decision. The typical optimization goal exists time delay [6], energy consumption [5], [11] and delay-energy consumption tradeoff [12], [13] neglecting the access costs caused by ECS access

The associate editor coordinating the review of this manuscript and approving it for publication was Ning Zhang.

delay and resource occupancy. Finally, for multi-user and multi-task offloading, the optimization goal usually manages to minimize the sum offloading cost. It probably occurs the scenario that one offloads complete tasks with good offloading performance, meanwhile, the others cannot offload any task, and this situation will result in unfair competition. This paper studies multi-user and multi-task offloading decisions based on unbalanced edge cloud, where multiple users share multiple ECS computing resources via multiple APs. Each mobile user processes multiple compute-intensive or delay-sensitive tasks, and offloading decisions involve AP and ECS selection. The main contributions can be summarized as follows.

1) For the unbalanced ECS deployment, the multi-user and multi-task offloading decision algorithm is studied. Multiple APs are not directly connected to the ECS, but multiple ECSs are shared by multi-hop/single-hop backhaul. This architecture is more practical than the ECS balanced deployment.
2) Different from existing research, not only taking MU-AP transmission delay and energy consumption in to consideration, but also the paper considers different service level agreements (SLA) [14], [15] between virtual operators and cloud service providers and thereof introduces the ECS access cost that characterizes ECS access delay [16] and resource occupancy to quantify the user experience of task offloading by getting the tradeoff among time delay, energy consumption and cost. The optimal ECS access cost can be an optimization target for offloading decision.
3) Both of the optimization problems that minimizing the sum offloading costs for all MUs (efficiency-based) and minimizing the maximal offloading cost per MU (fairness-based) are discussed. Two suboptimal algorithms based on linear relaxation are proposed respectively, and the performance of each algorithm is verified by numerical simulation. The former has higher resource utilization efficiency and the latter has better fairness among users.

II. SYSTEM MODEL AND OPTIMIZATION MODEL

A. SYSTEM MODEL

As shown in Fig.1, the MEC system model based on the unbalanced edge cloud is a three-tier topology composed of ECS-AP-MU, the sets of MUs, APs and ECSs in the system are denoted as $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}, \mathcal{B} = \{b_1, \ldots, b_{|\mathcal{B}|}\}$, $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ respectively. The ith MU's tasks set is denoted by $S_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,S_i}\}$, and the computing resources that the task $s_{i,k}$ are in demand are expressed as $r_{i,k}, i = 1, \ldots, |\mathcal{A}|, k = 1, \ldots, \mathcal{S}_i$. It is assumed that the calculate ability or battery capacity of the MU is limited so that all tasks should be offloaded to the ECSs via the APs and APs share the whole ECSs through backhaul networks. Different access of ECSs possesses disparate access delay and resource occupation so that different APs have different ECS access costs. The access cost of AP m accessing to ECS n is denoted as $\delta_{m,n}, m = 1, \ldots, |\mathcal{B}|, n = 1, \ldots, |\mathcal{C}|$, and the transmission delay and energy consumption of the ith MU's task $s_{i,k}$ to APm are denoted as $t_{i,k,m}$ and $e_{i,k,m}$, respectively. Even if different MUs access the same AP, their transmission delay and energy consumption are different due to channel conditions.

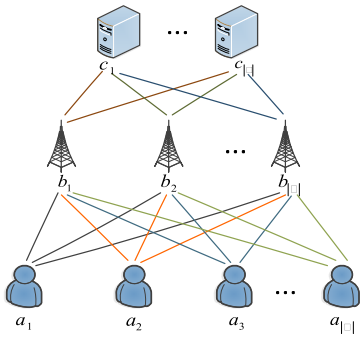

FIGURE 1. System model.

we assume that the number of orthogonal channels of AP m is $\mathcal{Q}_m$ and the computing resource of ECS n is R_n, so the offloading decision mainly depends on AP and ECS selection, denoted as $x_{i,k,m,n}$. More precisely, $x_{i,k,m,n} = 1$ means that the ith MU of kth task $s_{i,k}$ offloads to ECSn through AP m, and vice versa $x_{i,k,m,n} = 0$ implies no offloading. The offloading cost of task $s_{i,k}$ is defined as the weighted sum of transmission delay, energy consumption and ECS access cost, expressed as

$$u_{i,k} = \sum_{m=1}^{|\mathcal{B}|} \sum_{n=1}^{|\mathcal{C}|} x_{i,k,m,n} \left(\alpha_i t_{i,k,m} + \beta_i e_{i,k,m} + \gamma_i \delta_{m,n}\right) \quad (1)$$

where α_i, β_i, γ_i represent the weighting coefficients of transmission delay, energy consumption, and ECS access cost, respectively, reflecting the preference of MU i for task offloading to the corresponding overhead. The total cost for all tasks of MU i is expressed as:

$$U_i = \sum_{k=1}^{\mathcal{S}_i} u_{i,k} = \sum_{k=1}^{\mathcal{S}_i} \sum_{m=1}^{|\mathcal{B}|} \sum_{n=1}^{|\mathcal{C}|} x_{i,k,m,n} \left(\alpha_i t_{i,k,m} + \beta_i e_{i,k,m} + \gamma_i \delta_{m,n}\right) \quad (2)$$

B. OPTIMIZATION MODEL

The optimization model aiming to minimize the sum offloading costs of tasks of all MUs based on efficiency is expressed as:

$$\begin{aligned} \text{OP1}: \min & \sum_{i=1}^{|\mathcal{A}|} \sum_{k=1}^{\mathcal{S}_i} \sum_{m=1}^{|\mathcal{B}|} \sum_{n=1}^{|\mathcal{C}|} x_{i,k,m,n} (\alpha_i t_{i,k,m} + \beta_i e_{i,k,m} + \gamma_i \delta_{m,n}) \\ s.t.\ (a) & \sum_{i=1}^{|\mathcal{A}|} \sum_{k=1}^{S_i} \sum_{m=1}^{|\mathcal{B}|} x_{i,k,m,n} r_{i,k} \le \mathcal{R}_n, \\ & \forall n \in \{1, \ldots, |\mathcal{C}|\} \end{aligned}$$

$$
\begin{aligned}
&(b)\ \sum_{i=1}^{|\mathcal{A}|}\sum_{k=1}^{\mathcal{S}_i}\sum_{n=1}^{|\mathcal{C}|} x_{i,k,m,n} \le \mathcal{Q}_m,\\
&\forall m \in \{1,\ldots,|\mathcal{B}|\}\\
&(c)\ \sum_{m=1}^{|\mathcal{B}|}\sum_{n=1}^{|\mathcal{C}|} x_{i,k,m,n} = 1, \quad \forall i \in \{1,\ldots,|\mathcal{A}|\},\\
&k \in \{1,\ldots,\mathcal{S}_i\}\\
&(d)\ x_{i,k,m,n} \in \{0,1\}, \quad \forall i \in \{1,\ldots,|\mathcal{A}|\},\\
&k \in \{1,\ldots,\mathcal{S}_i\},\ m \in \{1,\ldots,|\mathcal{B}|\},\\
&n \in \{1,\ldots,|\mathcal{C}|\}
\end{aligned} \tag{3}
$$

where (a) indicates that the computing resources of all MUs of selected ECS cannot exceed the computing resources of ECS;(b) indicates that the number of MUs accessing the APs should not exceed the number of orthogonal channels; (c) indicates that a task can only be assigned to one ECS through one AP.

The objective of OP1 is to minimize the sum offloading cost of all MUs. However, if the full-task-offloading costs of some MUs are relatively low comparing with others, the case may be caused that one MU offloads its whole tasks and occupies amounts of access. This will result in unfair offloading decisions. In view of this, the optimization model based on fairness is formed:

$$
\begin{aligned}
\text{OP2}: \ &\min \max_i \{\eta_i \sum_{k=1}^{|S_i|}\sum_{m=1}^{M}\sum_{n=1}^{N} x_{i,k,m,n}\\
&(\alpha_i t_{i,k,m} + \beta_i e_{i,k,m} + \gamma_i \delta_{m,n})/\,|S_i|\}\\
&s.t.\ (a),(b),(c),(d)
\end{aligned} \tag{4}
$$

where the weight η_i reflects equitable degree between MUs. Obviously, the offloading cost of MU i becomes lower with the increasing number of η_i and the difference between OP2 and OP1 only exists in the objective function. It should be noted that the equitable degree is reflected at the user level, that is, the gap between the average task offloading costs between any two MUs is kept as small as possible.

III. EFFICIENCY-BASED OFFLOADING DECISION ALGORITHM

A. FEASIBILITY ANALYSIS

For OP1 and OP2, it is apparent that the feasible solution may be an empty set. As mentioned before, we assume that a task is only offloaded from one AP to one ECS and the AP and ECS meet resource constraints. In order to ensure that the OP1 or OP2 feasible solution is not empty, the following implicit constraints must be satisfied:

$$
\sum_{i=1}^{|\mathcal{A}|}\sum_{k=1}^{\mathcal{S}_i} r_{i,k} \le \sum_{n=1}^{|\mathcal{C}|} \mathcal{R}_n \tag{5}
$$

$$
\sum_{i=1}^{|\mathcal{A}|}\sum_{k=1}^{\mathcal{S}_i}\sum_{n=1}^{|\mathcal{C}|} x_{i,k,m,n} \le \sum_{m=1}^{|\mathcal{B}|} \mathcal{Q}_m \tag{6}
$$

The former indicates that the sum of computing resources required by all MUs does not exceed the amount of total computing resources that all ECSs can provide. The latter indicates that the number of tasks requesting services does not exceed the number of connections supported by the APs.

In fact, even if (5) and (6) are satisfied, the feasible set may still be an empty set since the computational resource demands of multiple tasks cannot be aligned on each ECS. Cite an instance, assume that two ECSs, $c_1$andc_2, are configured, and their computing resources can supply to the MUs are $R_1 = 3$and$R_2 = 4$. There are two tasks s_1, s_2 requiring computing resources $r_1 = 1, r_2 = 5$, respectively. Although $r_1 + r_2 \le R_1 + R_2$, only one task can be offloaded and OP1 still has no feasible solution. In order to avoid the above problems, we assume that the available resources of the APs and ECSs are sufficient to support all tasks in the system. For example, the system is equipped with a central cloud server with sufficient computing resources or a offloading decision algorithm with built-in user scheduling strategy in [6] to support the computations mentioned above.

B. LINEAR RELAXATION

The essence of OP1 is a combinatorial optimization problem, which is belong to NP-hard and can be solved by exhaustive search or derivation of the necessary and sufficient condition of the optimal solution. The former needs to traverse all feasible solutions, but each task has $|\mathcal{B}| \times |\mathcal{C}|$ possible offloading paths. If offloading tasks are $\Sigma_{i\in\mathcal{A}}\mathcal{S}_i$, the number of offloading path combination is equals to $(|\mathcal{B}| \times |\mathcal{C}|)^{\Sigma_{i\in\mathcal{A}}\mathcal{S}_i}$.

In particular, in the MEC system model with three ECSs, four APs and five MUs, if each MU demands two tasks, then the exhaustive search will traverse 12^{10} paths, leading to high time complexity. For the latter, due to the mutual coupling of AP and ECS and the existence of discrete variables, the necessary and sufficient condition for the optimal solution of OP1 cannot be derived. Therefore the linear relaxation is applied to OP1, and $x_{i,k,m,n}$ is relaxed to real $x_{i,k,m,n} \in [0,1]$. OP1 is transformed into OP1 (LR):

$$
\begin{aligned}
\text{OP1(LR)}: \ &\min \sum_{i=1}^{|\mathcal{A}|}\sum_{k=1}^{\mathcal{S}_j}\sum_{m=1}^{|\mathcal{B}|}\sum_{n=1}^{|\mathcal{C}|} x_{i,k,m,n}\pi_{i,k,m,n}\\
&s.t.\ (a),(b),(c), \quad x_{i,k,m,n} \in [0,1],\ \forall i \in \mathcal{A},\\
&k \in \mathcal{S}_i, \quad m \in \mathcal{B},\ n \in \mathcal{C}
\end{aligned} \tag{7}
$$

where, $\pi_{i,k,m,n} = \alpha_i t_{i,k,m} + \beta_i e_{i,k,m} + \gamma_i \delta_{m,n}$.

OP1 (LR) can use the interior point method to find the optimal solution $\hat{\mathbf{X}}$. If the components of $\hat{\mathbf{X}}$ are binary, then $\hat{\mathbf{X}}$ is also the optimal solution of OP1. Since OP1 still needs to satisfy (a), (b), (c).The original problem solution cannot be obtained by restoring the duality of $\hat{\mathbf{X}}$, that is, the optimal solution of OP1(LR) is the lower bound of OP1.

C. CENTRALIZED GREEDY ALGORITHM

A centralized greedy algorithm (CGA) is proposed to solve OP1 (LR). To simplify the description, symbol definitions are given.

l : number of iterations;

$\bar{\mathcal{A}}(l)$: user set of non-empty offloading task set after l iterations;

$\bar{\mathcal{S}}^i(l)$: local task set of MU i after iteration l times;

$\mathcal{Q}_m(l)$: number of allowed connections of AP m after iteration l times;

$\mathcal{R}_n(l)$: available computing resources of ECS n after iteration l times;

$\hat{\mathcal{B}}^{i,k}(l) = \{m|\mathcal{Q}_m(l) \geq 1, m \in \mathcal{B}\}$: the set of APs that can be accessed by task $s_{i,k} \in \bar{\mathcal{S}}^i(l)$ after iteration l times;

$\hat{\mathcal{C}}^{i,k}(l) = \{n|\mathcal{R}_n(l) \geq r_{i,k}, n \in \mathcal{C}\}$: the set of ECSs that can be chosen by task $s_{i,k} \in \bar{\mathcal{S}}^i(l)$ after iteration l times;

$\Delta^{i,k}(l) = \{\delta^{i,k}_{m,n} = \delta_{m,n}|m \in \hat{\mathcal{B}}^{i,k}(l), n \in \hat{\mathcal{C}}^{i,k}(l)\}$: ECS access cost for task $s_{i,k} \in \bar{\mathcal{S}}^i(l)$ after iteration l times.

CGA is executed by a virtual decision center (VDC). The VDC first collects information such as task computing resource demands, transmission delay, energy consumption of different APs, ECS access costs, AP connection constraints, and available computing resources of ECS before executing the CGA and then feeds the results back to the MUs. The core of the CGA is greedy thinking, that is the best offloading path (AP and ECS with minimum offloading cost) is assigned to task with the least offloading cost each time. Algorithm will take iteration and update information repeatedly until meet stop conditions: 1) Communication resources or computing resources cannot support more tasks; 2) All tasks have been offloaded. However, because the cost of task offloading is a trade-off between delay, energy consumption and cost, which depends on AP and ECS, greedy algorithm cannot be used directly to solve the problem. To this end, the following propositional design algorithm is introduced.

Proposition 1: If task $s_{i,k}, \forall i \in \mathcal{A}, k \in \mathcal{S}_i$ is offloaded through AP m, and $\mathcal{C}^{i,k}_m = \{n \,|R_n \geq r_{i,k}, n \in \mathcal{C}\}$, the optimal ECS of task $s_{i,k}$ is:

$$n_o = \arg\min_{n \in \mathcal{C}^{i,k}_m} \delta_{m,n} \tag{8}$$

The optimal ECS of task $s_{i,k}$ is ECS with the least access cost under the premise that the remaining computing resources are sufficient. For each offloading task, it is prior to execute the inner greedy algorithm (IGA) for the optimal offloading path. The input parameters are the set of APs and ECSs that can be accessed and the ECS access costs of the tasks, and the output is the optimal offloading decision.

IGA : $(m^{i,k}_o, n^{i,k}_o, u^{i,k}_o) = f(\hat{\mathcal{B}}^{i,k}(l), \hat{\mathcal{C}}^{i,k}(l), \Delta^{i,k}(l))$

1. **Input** $\hat{\mathcal{B}}^{i,k}(l), \hat{\mathcal{C}}^{i,k}(l), \Delta^{i,k}(l)$;
2. According to $\Delta^{i,k}(l)$, **search** $\hat{n}^{i,k}_m = \arg\min_{n \in \hat{\mathcal{C}}^{i,k}(l)} \delta^{i,k}_{m,n}$, $\forall k \in \bar{\mathcal{S}}^i(l)$;
3. **Calculate** $m^{i,k}_o = \arg\min_{m \in \hat{\mathcal{B}}^{i,k}(l)} u^{i,k}_m = \alpha_i t_{i,k,m} + \beta_i e_{i,k,m} + \gamma_i \delta^{i,k}_{m,\hat{n}^{i,k}_m}$, **renew** $n^{i,k}_o = \hat{n}^{i,k}_{m^{i,k}_o}, u^{i,j}_o = u^{i,j}_{m^{i,j}_o}$;
4. **Output** $(m^{i,k}_o, n^{i,k}_o, u^{i,k}_o)$.

In iteration l of GGA, the AP set, ECS set, and ECS access cost for each local task are updated based on available communication resources and computing resources, i.e.$\hat{\mathcal{B}}^{i,k}(l), \hat{\mathcal{C}}^{i,k}(l), \Delta^{i,k}(l)$. Next, the optimal offloading path for each offloading task is found by using IGA, i.e. $(m^{i,k}_o, n^{i,k}_o, u^{i,k}_o)$. Then, for each MU $i \in \bar{\mathcal{A}}(l)$, we search for the task with the lowest offloading cost. With the comparison of $\bar{\mathcal{A}}(l)$, the task k^* belonging to MU i^* is selected, and, meanwhile, the task offloads to the ECS n^* via the AP m^*. Finally, the remaining communication resources, computing resources, and MU local task sets are updated. The termination conditions are set to: 1) insufficient remaining computing resources; 2) insufficient remaining communication resources; 3) all tasks are offloaded.

CGA:

1. **Initialize**:$l = 0, \mathcal{Q}_m(l) = \mathcal{Q}_m, \mathcal{R}_n(l) = \mathcal{R}_n$, $\mathcal{A}(l) = \mathcal{A}, \bar{\mathcal{S}}^i(l) = \mathcal{S}_i$.
2. $i \in \bar{\mathcal{A}}(l)$, **execute**:
   2.1 $\forall k \in \bar{\mathcal{S}}^i(l)$, **renew**$\hat{\mathcal{B}}^{i,k}(l), \hat{\mathcal{C}}^{i,k}(l), \Delta^{i,k}(l)$.
   2.2 **Execute** the IGA,$(m^{i,k}_o, n^{i,k}_o, u^{i,k}_o)$ $= f(\hat{\mathcal{B}}^{i,k}(l), \hat{\mathcal{C}}^{i,k}(l), \Delta^{i,k}(l))$.
   2.3 **Calculate**: $i^* = \arg\min_{i \in \bar{\mathcal{A}}(l)} u^i_o$, **renew** $k^* = k^{i^*}_o$, $m^* = m^{i^*}_o, n^* = n^{i^*}_o, u^* = u^{i^*}_o$.
3. **Calculate**: $i^* = \arg\min_{i \in \bar{\mathcal{A}}(l)} u^i_o$, **renew** $k^* = k^{i^*}_o$, $m^* = m^{i^*}_o, n^* = n^{i^*}_o, u^* = u^{i^*}_o$.
4. If $\bar{\mathcal{S}}^{i^*}(l) = \Phi, \bar{\mathcal{A}}(l) = \bar{\mathcal{A}}(l)\ i^*$, then $\mathcal{R}_{n^*}(l) = \mathcal{R}_{n^*}(l) - r_{i^*,k^*}$, $\mathcal{Q}_{m^*}(l) = \mathcal{Q}_{m^*}(l) - 1, \bar{\mathcal{S}}^{i^*}(l) = \bar{\mathcal{S}}^{i^*}(l)\ k^*$.
5. If $\max_{n \in \mathcal{C}} \mathcal{R}_n(l) \leq \min_{i \in \bar{\mathcal{A}}(l), k \in \bar{\mathcal{S}}^i(l)} r_{i,k}$ or $\mathcal{Q}_m(l) < 1, \forall m \in \mathcal{B}$ or $\bar{\mathcal{A}}(l) = \Phi$, The algorithm ends; otherwise $l = l + 1$, jump to step2.

For example: we assume that two ECSsc_1, c_2 are configured and possess computation resources R_1, R_2.There are tasks s_1, s_2, s_3 requiring computation resources r_1, r_2, r_3 need to be offloaded. In addition, we assume:

$$r_1 \leq R_1, r_2 \leq R_2, R_1 - r_1 < r_2, R_1 - r_1 < r_3,$$
$$r_2 + r_3 \leq R_1, r_2 + r_3 \leq R_2, R_1 - (r_2 + r)_3 < r_1.$$

That is ECSs c_1, c_2 can support tasks s_1, s_2, s_3. If ECSc_1 is assigned to tasks_1, the ECS cannot support tasks_2 or task s_3. On the contrary, suppose ECS c_1 is assigned to task s_2 and task s_3, the ECS cannot support task s_1. In case the offloading costs that tasks s_1, s_2, s_3 access c_1, c_2 is $u^1_1, u^2_1, u^1_2, u^2_2, u^1_3, u^2_3$, respectively. We assume:

$$u^1_1 < u^2_1, u^1_2 < u^2_2, u^1_3 < u^2_3, u^1_1 < u^1_2, u^1_1 < u^1_3$$

The offloading decision of the CGA is described as follows, the task s_1 assign to ECS c_1, the tasks s_2 and s_3 assign to ECSc_2. Obviously, the offloading decision is feasible, and $u_{CGA} = u^1_1 + u^2_2 + u^2_3$, where u_{CGA} denoted as the sum offloading cost. Another feasible offloading decision is considered as follows, the task s_1 assign ECSc_2, the tasks s_2 and s_3 assigning ECSc_1, and $u' = u^2_1 + u^1_2 + u^1_3$.. In particular, if

$$(u^2_2 + u^2_3) - (u^1_2 + u^1_3) > (u^2_1 - u^1_1) > 0$$

In other words, $u' < u_{CGA}$ and it indicates that the CGA is a suboptimal algorithm.

The CGA only considers the offloading cost when makes the offloading decision, ignoring the influence of the computing resources, so that, resulting in a suboptimal solution. Therefore, the revised offloading cost for a task is defined as:

$$\hat{u}_{i,k} = u_{i,k}^{\epsilon} r_{i,k}^{\zeta} \tag{9}$$

In addition to the offloading costs, the revised offloading cost also includes the required computing resources, where the weights of original offloading cost and computing resource consumption are denoted as offloading decision parameters $\epsilon \geq 1$ and $\zeta \geq 1$, respectively. Based on (9), step 2.3 is modified to:

$$k_o^i = \arg\min_{k \in \bar{\mathcal{S}}^i(l)} \hat{u}_o^{i,k} \tag{10}$$

Hereto, the modified greedy algorithm (MGA) is proposed. It can be seen that the values of the offloading decision parameters ϵ and ζ will affect the offloading cost and computing resource consumption so as to further affect the performance of the algorithm. However, it is difficult to theoretically derive the optimal values of these two parameters. This paper will analyze the impact of offloading decision parameters on the performance of the algorithm through simulation.

Proposition 2: The upper bound of the complexity for CGA is:

$$O\left(\sum_{i\in\mathcal{A}} |\mathcal{S}_i| \left(|\mathcal{A}|^2 + |\mathcal{A}| \left(|\mathcal{S}_i|^2 + |\mathcal{S}_i| \left(|\mathcal{B}|^2 + |\mathcal{C}|^2\right)\right)\right) + \left(\sum_{i\in\mathcal{A}} |\mathcal{S}_i|\right)^2 + |\mathcal{C}|^2\right) \tag{11}$$

Proof: Firstly, in each iteration, the algorithm makes an offloading decision for one task and will not interrupt until the remaining communication resources or computing resources are insufficient to support more tasks or all tasks have been offloaded. In other words, the algorithm will stop in case that no more than $\Sigma_{i\in\mathcal{A}} |\mathcal{S}_i|$ iteration. Secondly, the operations contain steps 2.2, 2.3, 3 and 5 in each iteration. Step 2.2 is performed at the task level with a complexity of $O(|\mathcal{B}|^2 + |\mathcal{C}|^2)$; Step 2.3 is performed at the user level, and the maximum complexity is $O(|\mathcal{S}_i|^2)$; Step 3 is performed at the system level with a complexity of $O(|\mathcal{A}|^2)$; For Step 5, although we check the condition in each iteration, it only needs to be calculated once and then updated with the complexity of $(\Sigma_{i\in\mathcal{A}} |\mathcal{S}_i|)^2 + |\mathcal{C}|^2$. One thing should be point out that other operations have low computational complexity, so the complexity of CGA is analyzed only from the perspective of sorting.

Proposition 3: $u_{m,n}^{i,k}(l)$ represents the offloading cost of the task $s_{i,k}$ of MU i offloading to the ECS n_1 through AP m_1 in the lth iteration. $u_{m_1,n_1}^{i_1,k}(l_1) = \min_{m\in\mathcal{B},n\in\mathcal{C}} u_{m,n}^{i_1,k}$ is defined as the optimal offloading cost of the task $s_{i_1,k}$ offloading to the ECS n_1 through the AP m_1 in the l_1th iteration. Similarly, $u_{m_2,n_2}^{i_2,k}(l_2) = \min_{m\in\mathcal{B},n\in\mathcal{C}} u_{m,n}^{i_2,k}$ is defined as the optimal offloading cost of the task $s_{i_2,k}$ offloading to the ECS n_2 through the AP m_2 in the l_2th iteration. If $u_{m_1,n_1}^{i_1,k}(l_1) \leq u_{m_2,n_2}^{i_2,k}(l_2)$, then $l_1 \leq l_2$, which means that the offloading decision of task $s_{i_1,k}$ takes precedence over tasks$s_{i_2,k}$.

Proposition 4: Offloading cost $\hat{u}^{i,k}(l)$ of task $s_{i,k}$ at the lth iteration is a non-decreasing function for l.

Proof: In the CGA, the offloading decision problem of task $s_{i,k}$ in the lth iteration is given:

$$\begin{aligned} \text{OP-CGA}(l): \hat{u}^{i,k}(l) &= \min_{x_{i,k,m,n}} u_{m,n}^{i,k}(l) \\ s.t.\ & x_{i,k,m,n} \in \{0,1\}, \quad m \in \hat{\mathcal{B}}\mathcal{A}^{i,k}(l),\ n \in \hat{\mathcal{C}}^{i,k}(l), \\ & l = 1, \ldots, \Sigma_{i\in\mathcal{A}} |\mathcal{S}_i| \end{aligned} \tag{12}$$

where $\hat{\mathcal{B}}^{i,k}(l)$ and $\hat{\mathcal{C}}^{i,k}(l)$ are accessible sets of AP and ECS sets for task $s_{i,k}$ in the lth iteration, respectively. By analyzing the CGA, we found that $\mathcal{Q}_m(l), m \in \mathcal{B}$ and $\mathcal{R}_n(l), n \in \mathcal{C}$ are non-increasing functions for l. That is, with the increase of l, at least one AP's communication resources and one ECS's computing resources should be decrease. In the feasible solution set $\mathcal{F}_{CGA}(l)$ of OP-CGA(l), if $l_1 < l_2$, then $\mathcal{F}_{CGA}(l_2) \subseteq \mathcal{F}_{CGA}(l_1)$. As l increases, the feasible solution set becomes smaller, and the optimal value of OP-CGA(l) will become larger [17], i.e. $\hat{u}^{i,k}(l_1) \leq \hat{u}^{i,k}(l_2)$.

According to Proposition 3, if the optimal offloading cost of a task is less than other tasks, the offloading decision will be completed first. Therefore, if all tasks of some MUs have smaller offloading costs, the offloading decisions for these tasks will take precedence over tasks of other MUs. According to Proposition 4, this will result in higher offloading efficiency. However, although CGA improves offloading efficiency, it may lead to unfairness between MUs.

IV. FAIRNESS-BASED OFFLOADING DECISION ALGORITHM

The combination optimization problem OP2 based on fairness is also NP-hard and cannot be solved directly. To solve OP2, we introduce a new variable y into the equivalent problem OP3 as follows:

$$\begin{aligned} \text{OP3}: \min_{x_{i,k,m,n},y}\ & y \\ s.t.\ & \sum_{k=1}^{\mathcal{S}_i} \sum_{m=1}^{|\mathcal{B}|} \sum_{n=1}^{|\mathcal{C}|} x_{i,j,m,n} \\ & (\alpha_i t_{i,k,m} + \beta_i e_{i,k,m} + \gamma_i \delta_{m,n}) \leq y\,|\mathcal{S}_i|/\eta_i, \\ & \forall i \in \mathcal{A} \\ & (a), (b), (c), (d) \end{aligned} \tag{13}$$

where the binary variable $x_{i,k,m,n}$ is relaxed to a real number $x_{i,k,m,n} \in [0,1]$ based on linear relaxation, and OP3 is transformed into OP3 (LR):

$$\begin{aligned} \text{OP3(LR)}: \min_{x_{i,k,m,n},y}\ & y \\ s.t.\ & (a), (b), (c) \\ & x_{i,k,m,n} \in [0,1], \quad \forall i \in \mathcal{A},\ k \in \mathcal{S}_i, \\ & m \in \mathcal{B},\ n \in \mathcal{C},\ y \geq 0. \end{aligned} \tag{14}$$

A fairness based greedy algorithm (FGA) is proposed to solve OP3 (LR). The algorithm proposes a fairness-based MU scheduling strategy. It is built into the offloading decision algorithm. The priority of the MU i at the lth iteration is defined as:

$$\chi^i(l) = \left(Y\left|\mathcal{S}_i\right|/\eta_i - \sum\nolimits_{k\in\hat{\mathcal{S}}^i(l)} u^{i,k}\right)/\left|\bar{\mathcal{S}}^i(l)\right| \tag{15}$$

where $Y = \max_{i\in\mathcal{A},k\in\mathcal{S}_i,m\in\mathcal{B}} t_{i,k,m} + \max_{i\in\mathcal{A},k\in\mathcal{S}_i,m\in\mathcal{B}} e_{i,k,m} + \max_{m\in\mathcal{B},n\in\mathcal{C}} \delta_{m,n}$, $\hat{\mathcal{S}}^i(l)$ and $\bar{\mathcal{S}}^i(l)$ are the offloading task set and the local task set of MU i at the lth iteration, respectively. $u^{i,k}$ is the offloading cost of task $s_{i,k}$ in $\hat{\mathcal{S}}^i(l)$, $Y\left|\mathcal{S}_i\right|/\eta_i$ represents the offloading decision target of MU i, $\Sigma_{k\in\hat{\mathcal{S}}_i(l)} u^{i,k}$ represents the offloading cost of the MU i offloaded task. For MU i, if $\chi^i(l)$ is small, the task constraints is tight in $\bar{\mathcal{S}}_i(l)$. From Proposition 4, the offloading cost is a non-decreasing function of l. Therefore, in order to meet the offloading cost and constraints of the MU, we should select the MU offloading with the smallest $\chi^i(l)$ at the lth iteration. The priority of MU i is negatively correlated with $\chi^i(l)$.

FGA:

1. **Initialize**: $l = 0$, $Y = \max_{i\in\mathcal{A},k\in\mathcal{S}_i,m\in\mathcal{B}} t_{i,k,m} + \max_{i\in\mathcal{A},k\in\mathcal{S}_i,m\in\mathcal{B}} e_{i,k,m} + \max_{m\in\mathcal{B},n\in\mathcal{C}} \delta_{m,n}$ and $\mathcal{Q}_m(l) = \mathcal{Q}_m$, $\mathcal{R}_n(l) = \mathcal{R}_n$, $\bar{\mathcal{A}}(l) = \mathcal{A}$, $\bar{\mathcal{S}}^i(l) = \mathcal{S}_i$, $\hat{\mathcal{S}}^i(l) = \emptyset$
2. $i \in \bar{\mathcal{A}}(l)$, **calculate** $\chi^i(l)$ and $i^* = \arg\min_{i\in\bar{\mathcal{A}}(l)} \chi^i(l)$;
3. MU $i = i^*$, **execute:**
   3.1 $\forall j \in \bar{\mathcal{S}}^i(l)$, **calculate** $\hat{\mathcal{B}}^{i,k}(l)$, $\hat{\mathcal{C}}^{i,k}(l)$, $\Delta^{i,k}(l)$;
   3.2 **Implement** the IGA algorithm, $(m_o^{i,k}, n_o^{i,k}, u_o^{i,k}) = f(\hat{\mathcal{B}}^{i,k}(l), \hat{\mathcal{C}}^{i,k}(l), \Delta^{i,k}(l))$;
   3.3 **Calculate** $k_o^i = \arg\min_{n\in\bar{\mathcal{S}}^i(l)} u_o^{i,k}$, **renew** $m_o^i = m_o^{i,k_o^i}$, $n_o^i = n_o^{i,k_o^i}$, $u_o^i = u_o^{i,k_o^i}$
4. **Renew** $k^* = k_o^{i^*}$, $m^* = m_o^{i^*}$, $n^* = n_o^{i^*}$, $u^* = u_o^{i^*}$, that is, the task k^* of MU i^* is offloaded to ECS n^* through AP m^*;
5. **Renew** $\mathcal{R}_{n^*}(l) = \mathcal{R}_{n^*}(l) - r_{i^*,k^*}$, $\mathcal{Q}_{m^*}(l) = \mathcal{Q}_{m^*}(l) - 1$, $\bar{\mathcal{S}}^{i^*}(l) = \bar{\mathcal{S}}^{i^*}(l)\,k^*$, $\hat{\mathcal{S}}^{i^*}(l) = \hat{\mathcal{S}}^{i^*}(l)\,k^*$.
   If $\bar{\mathcal{S}}^{i^*}(l) = \emptyset$, $\bar{\mathcal{A}}(l) = \bar{\mathcal{A}}(l)\,i^*$;
   If $\max_{n\in\mathcal{C}} \mathcal{R}_n(l) \le \min_{i\in\bar{\mathcal{A}}(l),k\in\bar{\mathcal{S}}^i(l)} r_{i,k}$ or $\mathcal{Q}_m(l) < 1, m \in \mathcal{B}$ or $\bar{\mathcal{A}}(l) = \emptyset$, the algorithm stops; otherwise $l = l+1$, jump to step 2.

The scheduling order of the MU can be determined based on the priority. When the MU has multiple tasks to offload, in order to complete the offloading decision, the task of performing the offloading needs to be further determined. Similar to the CGA, the FGA is based on greedy thinking and selects the task with the lowest MU i offloading cost. In the lth iteration, the scheduling order of MU is first determined, that is, i^* is determined, but $\arg\min_{i\in\bar{\mathcal{A}}(l)} \mathcal{X}^i(l)$ may be expressed as multiple MUs. For example, when $l = 0$, $\eta_{i_1} = \eta_{i_2}$ and $\forall i_1 \ne i_2 \in \mathcal{A}$, firstly, one MU is randomly selected. Then we determine the offloading task of MUi^*, and find the optimal offloading task of i^* based on greed. $(i^*, j^*, m^*, n^*, u^*)$ represent the offloading user, offloading task, offloading path and offloading cost, respectively. Finally update status. The stop condition of FGA is the same as CGA.

Proposition 5: The upper bound of the algorithm complexity of FGA as follows:

$$O\left(\sum\nolimits_{i\in\mathcal{A}} \left|\mathcal{S}_i\right| \left(\left|\mathcal{A}\right|^2 + \left|\mathcal{S}_i\right|^2 + \left(\left|\mathcal{B}\right|^2 + \left|\mathcal{C}\right|^2\right)\left|\mathcal{S}_i\right|\right) + \left(\sum\nolimits_{i\in\mathcal{A}} \left|\mathcal{S}_i\right|\right)^2 + \left|\mathcal{C}\right|^2\right) \tag{16}$$

Proof: Firstly, the FGA only makes offloading decisions for one task each iteration, where the most iteration times is $\Sigma_{i\in\mathcal{A}} \left|\mathcal{S}_i\right|$. Secondly, the operations contain steps 2, 3.2, 3.3 and 5 in each iteration. Step 2 is performed at the system level with a complexity of $O(|\mathcal{A}|^2)$; Step 3.2 is performed at the task level, and the complexity of sorting operation of each task $k \in \bar{\mathcal{S}}^i(l)$ is $O(|\mathcal{B}|^2 + |\mathcal{C}|^2)$; Step 3.3 is performed at the user level, and the maximum complexity is $O(|\mathcal{S}_i|^2)$; Step 5 only needs to be executed once, and the complexity is $(\Sigma_{i\in\mathcal{A}} |\mathcal{S}_i|)^2 + |\mathcal{C}|^2$.

V. SIMULATION ANALYSIS

In this section, we evaluate the proposed offloading decision algorithm by simulation. The scene parameters are randomly generated, including the number of MUs, the number of tasks per MU, the computing resources required for each task, the delay and energy consumption of the task from the MU to the AP, and the ECS access cost. We use the ELR algorithm and the FLR algorithm [18] provided by the CVX tool as a benchmark for offloading cost and fairness. Unless otherwise stated, all of the following simulation results are averaged over 1000 independent scenarios.

The Jain index evaluation algorithm based on the fairness is defined as follows [19]:

$$J(\mathbf{U}) = \frac{1}{n}\frac{\left(\sum_i U_i\right)^2}{\sum_i U_i^2} \tag{17}$$

The Fairness is positively correlated with the Jain index. Where $J(\mathbf{U}) \in [0, 1]$.

A. ALGORITHM PERFORMANCE BOUNDARY

Fig.2 and Fig.3 show the offloading cost and fairness curves of the four algorithms (CGA, FGA, ELR, and FLR) with the average computing resource required. The scene parameters

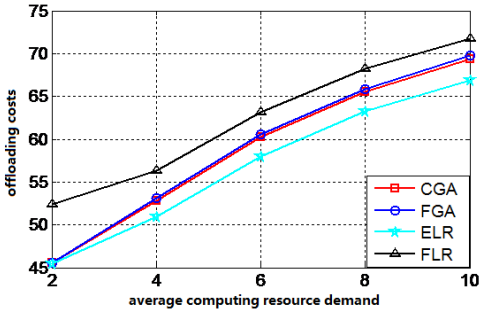

FIGURE 2. The offloading costs of different algorithm are compared with the average computing resource demand.

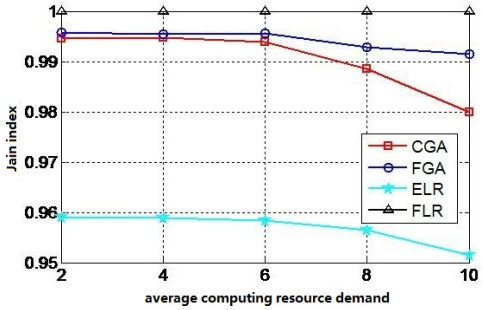

FIGURE 3. The fairness of different algorithm is compared with the average computing resource demand.

are: $|\mathcal{C}| = 2$, $|\mathcal{B}| = 3$, $|\mathcal{A}| = 5$, $\mathcal{S}_i = 3$, $\forall i \in \mathcal{A}$, $t_{i,k,m} \in [2, 6]$, $e_{i,k,m} \in [2, 6]$ and $\delta_{m,n} \in [1, 6]$.

As shown in Fig.2, we can find that, with the average computing resource required of the task increases, the offloading cost increases. The reason is that the minimum offloading cost path for the task is reduced. The offloading costs from high to low are ELR, CGA, FGA and FLR. In addition, when the average computing resource required is small, such as $r_{i,k} \leq 2$, the offloading costs of CGA, and FGA converge to the ELR performance limit. With the average computing resource demand increasing, the gap of offloading cost between CGA, FGA and ELR increase. The reason is that the task average computing resource demand is increases when the same low offloading cost path and number of tasks that ECS can support become large. Fig.3 shows that FLR has the best fairness, followed by FGA and CGA, and ELR has the worst fairness. In addition, as the average computing resource required increases, the fairness gap of the algorithm will increase, which is consistent with Fig.3. Fig.2 and Fig.3 show that ELR and FLR are upper bounds of efficiency and fairness for other algorithms, respectively.

B. COMPARISON OF CGA AND MGA

Fig.4 illustrates the offloading cost of CGA and MGA according to ϵ, and the scene parameters are the same as in Fig.2, $\eta = 3$. At $\epsilon \geq \bar{\epsilon} = 5.2$, the offloading cost of MGA is lower than CGA, which is consistent with the previous analysis. Further simulation shows that $\bar{\epsilon}$ depends on η and system configuration.

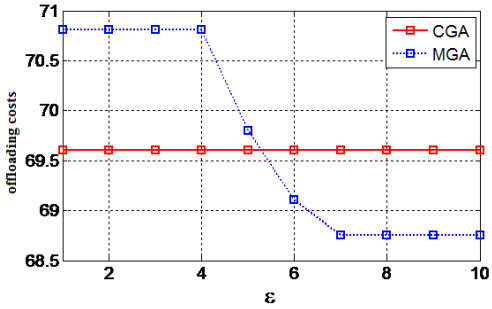

FIGURE 4. ϵ impact on offloading costs.

C. COMPARISON OF CGA AND FGA

Fig.5 and Fig.6 show changes in offloading costs and Jain index with each user task, respectively, where $|\mathcal{C}| = 4$, $|\mathcal{B}| = 8$, $|\mathcal{A}| = 20$. With the number of tasks of per MU increasing, the computing resource demand increases, and the offloading costs of CGA and FGA are consistent with Fig.3, and the fairness of FGA is better. The reason is that the user scheduling priority of (15) is adopted, and if the number of tasks of each MU is increased, the degree of freedom of scheduling is also increased, and the fairness is better.

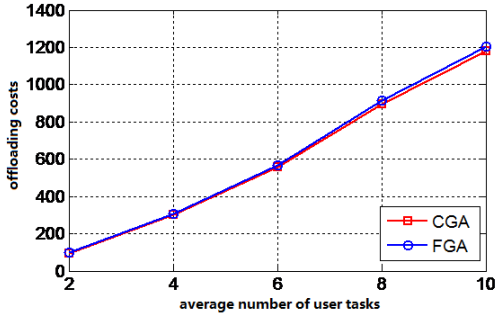

FIGURE 5. The impact of the average number of user tasks on the efficiency of CGA and FGA.

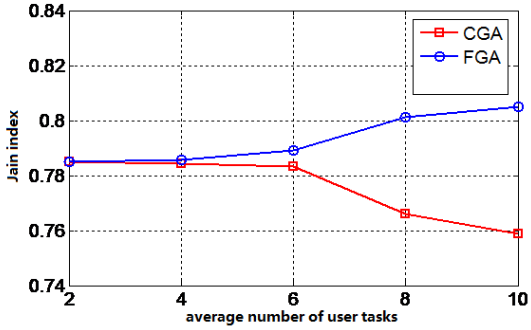

FIGURE 6. The impact of the average number of user tasks on the fairness of CGA and FGA.

Fig.7 shows the curve of task offloading rate with the number of MU when the system computing resources are limited, where $|\mathcal{C}| = 2$,$|\mathcal{B}| = 4$,$\mathcal{S}_i \in [2, 5]$. The offloading rate is defined as the number of tasks that were successfully offloaded in the total number of tasks. As the fig.7 shows, with the number of MU increasing, the offloading rate

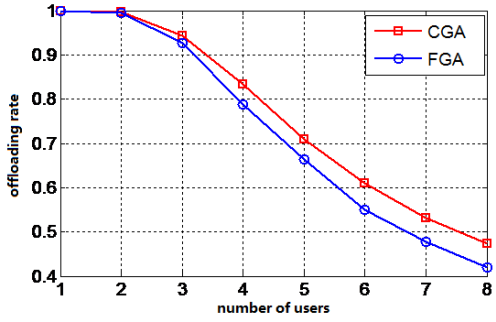

FIGURE 7. Impact of number of users on CGA and FGA offloading rate.

decreases. When the computing resources is limited, only some tasks can perform offloading computing. In addition, the gap of offloading rate between the two algorithms(CGA and FGA) increases, indicating that CGA resource utilization efficiency is better than FGA.

VI. CONCLUSION

In this paper, we studied the multi-user and multi-task offloading decision algorithm based on unbalanced edge cloud, and proposed a new criteria for task offloading performance evaluation, which is based on the tradeoff among time delay-energy consumption-cost. Through joint AP access and ECS selection, the optimization problems of minimizing the sum offloading cost of all MUs (based on efficiency) and minimizing the maximum offloading cost per MU (based on fairness) are discussed. The centralized algorithms, CGA, MGA based on efficiency, and FGA fairness-based, are proposed. With the bases of ELR and FLR, the simulation results show that along with the increase of average computing resource demands, the gap between CGA and FGA offloading cost (fairness) and ELR (FLR) is widening owing that the same low-cost offloading path and the number of tasks that ECS can support are reduced. The simulation of CGA and MGA shows that when the offloading decision parameter $\epsilon \geq 5.2$, the offloading cost of MGA is lower than CGA, which verifies the suboptimality of CGA. The comparative simulation between CGA and FGA shows that the fairness of CGA reduces when the amount of user tasks increases, whereas that of FGA decreases because of the increase of scheduling freedom of FGA. When the number of users increases, the offloading rate is decreased. While the offloading rate of CGA is always greater than FGA, and the gap increases with the increase of the number of users, indicating that the resource utilization efficiency of CGA is better than FGA.

REFERENCES

[1] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing a key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper 11, Sep. 2015, pp. 1–16, vol. 11, no. 11.

[2] 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.

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

[4] 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.

[5] 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.

[6] X. Wang, J. Wang, X. Wang, and X. Chen, "Energy and delay tradeoff for application offloading in mobile cloud computing," *IEEE Syst. J.*, vol. 11, no. 2, pp. 858–867, Jun. 2017.

[7] 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.

[8] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[9] 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.

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

[11] J. Cheng, Y. Shi, B. Bai, and W. Chen, "Computation offloading in cloud-RAN based mobile cloud computing system," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

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

[13] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[14] N. C. Luong, P. Wang, D. Niyato, Y. Wen, and Z. Han, "Resource management in cloud networking using economic analysis and pricing models: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 954–1001, 2nd Quart., 2017.

[15] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 358–380, 1st Quart., 2015.

[16] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct./Dec. 2015.

[17] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr./May 2014, pp. 352–357.

[18] M. Grant and S. Boyd. (Apr. 2011). *Matlab Software for Disciplined Convex Programming, Version 1.21*. [Online]. Available: http://stanford.edu/boyd/cvx

[19] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, "An axiomatic theory of fairness in network resource allocation," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

WEN-JIANG FENG received the Ph.D. degree in electrical engineering from Chongqing University, Chongqing, China, in 2000, where he is currently a Professor with the College of Microelectronics and Communication Engineering. His research interests include all aspects of MIMO communication including limited feedback techniques, antenna design, interference management and full-duplex communication, cognitive radio, special mobile communication systems, and emergency communication. He is also a Peer Review Expert of the Natural Science Foundation of China and a Senior Member of the China Institute of Communications. He also serves as an Editorial Board Member of Data Communication, China.

CHONG-HAI YANG received the B.S. degree in electronic information engineering from China West Normal University, Nanchong, China, in 2018. He is currently pursuing the master's degree in communication engineering, Chongqing University, China. His current research interests include mobile edge computing and interference alignment.

XIAO-SHAN ZHOU received the B.S. degree in microelectronics and communication engineering from Chongqing University, Chongqing, China, in 2015, where she is currently pursuing the master's degree in communication engineering. Her current research interests include mobile edg computing, ultra dense networks, and massive MIMO communication.