Joint Radio and Computation Resource Management for Low Latency Mobile Edge Computing

Qiang Liu, Tao Han
The University of North Carolina at Charlotte
Charlotte, NC. United States
Email: \{qliu12, tao.han\}@uncc.edu

Nirwan Ansari
New Jersey Institute of Technology
Newark, NJ. United States
Email: nirwan.ansari@njit.edu

Abstract—Mobile edge computing (MEC) is a new networking paradigm that enables low-latency computation offloading for compute-intensive mobile applications. The dynamic wireless channel, non-uniform spatiotemporal traffic, and limited computation resources impair the service latency of mobile edge computing. Therefore, jointly managing radio and computation resources is needed to achieve low latency MEC. In this paper, we propose a joint radio and computation resource management (iRAR) algorithm which minimizes users’ service latency by optimizing the uplink transmission power, receive beamforming, computation task assignment, and computation resource allocation. We compare the performance of the proposed algorithm with three different algorithms and demonstrate that the iRAR algorithm reduces up to 52% average service latency as compared to the other algorithms.

Index Terms—Mobile edge computing, computation offloading, beamforming, task assignment

I. INTRODUCTION

The rapid development of mobile techniques enable various applications, e.g., mobile augmented reality (MAR), virtual reality (VR), Internet of Things (IoT) [1]. These applications are compute-intensive and have stringent latency requirements. As a result, these applications are unsuitable to be executed in resource and battery-limited mobile devices. Mobile cloud computing (MCC) [2] is hence proposed to enable mobile devices to offload compute-intensive tasks to powerful cloud servers. Various computation offloading systems and frameworks have been proposed to improve the performance of mobile applications, e.g., MAUI [3], CloneCloud [4], ThinkAir [5].

The long distance between mobile users and cloud servers, however, introduces additional delay, which may be intolerable for some latency-sensitive applications [6]. To address this problem, mobile edge computing (MEC) is proposed as a new networking paradigm in which computing nodes are placed in close proximity, i.e., at the edge of networks, to mobile devices to provide highly responsive cloud services [7].

The service latency experienced by mobile users can be further reduced by utilizing the computation resources at the edge of networks. It is, however, not trivial to place cloud servers at the edge of networks. Edge servers usually have less computation resources than cloud servers. A single edge server can be easily overloaded by excessive service requests [8]. Hence, multiple edge servers are required to perform the computation tasks for mobile users. Meanwhile, the distribution of mobile users exhibits high temporal and spatial diversity [9]. The non-uniform user distribution may incur imbalanced workloads among edge servers, which impairs the performance of mobile edge computing in terms of the service latency [10]. Meanwhile, wireless communication channels are highly dynamic, which may result in considerable variations of user data rates and service latencies. Therefore, it is essential to jointly optimize radio and computation resource in order to achieve low latency MEC.

In this paper, we investigate the radio and computation resource management problem for delay-sensitive applications in mobile edge networks with multiple edge servers. We propose to jointly optimize the task assignment, radio and computation resource allocation to achieve low-latency MEC. The overall latency experienced by mobile users is characterized by modeling the wireless latency, network latency, and computation latency. Then, we formulate the optimization problem which minimizes the sum service latency experienced by users by optimizing the uplink transmission power, receive beamforming, task assignment and computation resource allocation. We develop a joint radio and computation resource management (iRAR) algorithm that solves the problem based on the block coordinate descent method. We evaluate the performance of the iRAR algorithm and validate its advantages over other algorithms through extensive simulations.

The remainder of this paper is organized as follows. Section II briefly reviews the related work. Section III presents the system model and problem formulation. Section IV develops the iRAR algorithm. Section V evaluates the performance of the proposed algorithm via extensive simulations. Section VI concludes the paper.

II. RELATED WORK

In mobile edge computing, compute-intensive mobile applications are usually offloaded to edge servers to reduce the computational latency. Hence, the joint radio and computation resource management problem is related to the computation offloading problem [11], [12]. There are many existing works on designing computation offloading schemes in mobile cloud computing as well as mobile edge computing. Huang et al.
[13] proposed an adaptive computation offloading algorithm to reduce the energy consumption of mobile devices while satisfying the applications’ latency constraints. The algorithm dynamically offloads parts of an application’s computation to a server according to wireless channel conditions. Zhang et al. [14] proposed a theoretical framework to save energy on mobile devices. In the framework, the CPU frequency of mobile devices can be reconfigured, and users’ data rates can be adapted according to the stochastic wireless channel conditions. Sardellitti et al. [15] proposed a joint optimization of transmit precoding and computation resource allocation algorithm to minimize the users’ energy consumption with the latency constraints in multicell MIMO system. You et al. [16] proposed a multiuser mobile edge computation offloading (MECO) system to reduce the energy consumption of mobile devices. In the system, the computation tasks are offloaded to cloud servers according to a priority function defined based on the users’ channel conditions and energy consumption. Chen et al. [17] formulated the distributed computation offloading problem as a multiuser game and designed a distributed computation offloading algorithm to achieve the Nash equilibrium of the game. These computation offloading solutions assume that all computation tasks are offloaded to the same cloud/edge server.

Pang et al. [18] proposed a latency-driven cooperative task computing algorithm to achieve a low latency computation offloading by jointly optimizing the cooperative computing nodes and near-range communication in fog radio access networks. Tong et al. [8] proposed a hierarchical edge cloud architecture to handle the non-uniform traffic among the geo-distributed edge servers and designed a task assignment algorithm to minimize the latency of users by optimizing the task placement and computation resource allocation. Tran et al. [19] minimized the latency and energy consumption of users by jointly optimizing the task offloading decision, uplink transmission power and computational resource allocation in OFDMA based mobile edge networks. However, their algorithms cannot guarantee that all users’ latency requirements are satisfied. Moreover, they did not consider using the multiple input multiple output (MIMO) techniques to reduce users’ latency.

III. SYSTEM MODEL

In this section, we describe the system model and formulate the joint radio and computation resource optimization problem. We consider a mobile edge network with $K$ users, $L$ base stations (BSs) and $N$ edge servers. Denote $K$, $L$ and $N$ as the set of users, BSs and servers, respectively. Each BS has $M$ antennas, and each user has one antenna. There is a centralized controller which manages the radio and computation resources in mobile edge networks. We assume that the information needed for the resource management, e.g., users’ latency tolerance and wireless channel conditions, can be obtained by the controller.

A. Wireless Communication Model

We consider the wireless network which adopts the time-division duplex (TDD) technique to separate its uplink and downlink transmissions. We denote $H^k \in \mathbb{C}^{1 \times M}$ as the downlink channel between the $k$th user and the $l$th BS, and $H = [H^1 \ldots H^K] \in \mathbb{C}^{LM \times K}$ as the downlink channel matrix between all users and BSs. The channel reciprocity is assumed so that the uplink channel matrix is $H_T \in \mathbb{C}^{LM \times K}$, where $(\cdot)^T$ is the transpose operator.

Then, the uplink received signal at the $l$th BS, denoted as $y_l$, can be written as

$$y_l = (H^k)^T \sqrt{p_k} s_k + \sum_{j \neq k, j \in K} (H^j)^T \sqrt{p_j} s_j + n_l, \forall l \in L,$$

where $p_k$ is the transmission power of the $k$th user, and $n_l$ is the received additive white Gaussian noise by the $l$th BS. Denote $\mathcal{P} = \{p_k \mid k \in K\}$ as the set of all users’ uplink transmission power.

We introduce $v^k_l \in \mathbb{C}^{1 \times M}$ as the uplink beamforming of the $k$th BS for the $l$th user and define $v^k = [v^1_k, v^2_k, \ldots, v^K_k]$. Denote $V = \{v_k \mid k \in K\}$ as the set of receive beamforming vectors. Here, we assume that an individual user has a single data stream, and the user’s signal independently and identically follows the distribution defined as $CN(0, 1)$. Therefore, with the linear receive beamforming scheme on the BSs, the achievable uplink data rate for the $k$th user can be written as

$$R_k = B \log \left(1 + \frac{p_k \|v_k (H^k)^T\|_2^2}{\sum_{j \neq k, j \in K} p_j \|v_k (H^j)^T\|_2^2 + \|v_k\|_2^2 \sigma^2}\right),$$

where $H^k = [H^k_1, H^k_2, \ldots, H^k_L] \in \mathbb{C}^{1 \times LM}$ is the channel matrix between the $k$th user and all BSs which characterizes the channel conditions; $\sigma^2$ is the noise power, $B$ is the system bandwidth, and $\|\cdot\|_2^2$ is the square of the $l_2$-norm.

B. Latency Model

In mobile edge computing, the user’s computation tasks are offloaded to edge servers via BSs. The experienced latency of the $k$th user can be defined as

$$T_k = T_k^w + T_k^d + T_k^p,$$

where $T_k^w$ is the wireless latency incurred by sending the data of the computation task from the $k$th user to a BS, $T_k^d$ is the network latency caused by transferring the $k$th user’s data from a BS to the assigned edge server, and $T_k^p$ is the computation latency for performing the task on the edge server. Since the data sizes of computation results are usually small, we do not model the latency caused by transmitting the results [15].

1) Wireless Latency Model: The wireless latency is determined by the data size of a user’s computation task and wireless data rate. Denote $d_k$ as the data size of the computation task of the $k$th user, then the wireless latency experienced by the $k$th user is modeled as

$$T_k^w = \frac{d_k}{R_k},$$

(4)
2) Network Latency Model: Denote $a_{k,n} \in \{0, 1\}$ as the task assignment indicator whether the $k$th user is served by the $n$th server. Denote $A = \{a_{k,n} | k \in K, n \in N\}$ as the set of users’ task assignments. Here, we set $a_{k,n}$ as a binary variable to impose a user to be served by only one server at a time. Let $\tau_{k,n}$ be the network latency between the $k$th user’s associated BS and the $n$th server, e.g., the data transportation latency from the BS to the assigned edge server. Hence, the network latency of the $k$th user can be expressed as

$$T_k^A = \sum_{n \in N} a_{k,n} \tau_{k,n}. \quad (5)$$

3) Computation Latency Model: The computation latency is closely related to the computational complexity of a user’s task and available computation resources on servers [15]. Let $c_k$ be the required computation resources for performing the $k$th user’s computation task. We define $f_{k,n}$ as the allocated computation resources, e.g., CPU cycles/second or GFLOPS (Giga Floating-point Operations Per Second), to the $k$th user by the $n$th server. Denote $F = \{f_{k,n} | k \in K, n \in N\}$ as the set of computation resource allocations. Therefore, the computation latency experienced by the $k$th user can be modeled as

$$T_k^C = \sum_{n \in N} a_{k,n} \frac{c_k}{f_{k,n}}. \quad (6)$$

Based on the above analytical model, the overall service latency experienced by the $k$th user is

$$T_k = \frac{d_k}{R_k} + \sum_{n \in N} a_{k,n} \left( \tau_{k,n} + \frac{c_k}{f_{k,n}} \right). \quad (7)$$

C. Problem Formulation

We aim to minimize the sum service latency experienced by users through joint optimization of the uplink transmission power $P$, receive beamforming $V$, task assignment $A$ and computation resource allocation $F$. Meanwhile, the practical constraints, such as the maximum transmission power and the maximum latency tolerance of users, are enforced. The optimization problem is formulated as

$$\mathcal{P}_0 : \min_{\{A,P,V,F\}} \quad T = \sum_{k \in K} T_k$$

$$\text{s.t.} \quad C_1 : \quad T_k \leq T_{k,\text{max}}, \forall k \in K,$$

$$C_2 : \quad p_k \leq P_{k,\text{max}}, \forall k \in K,$$

$$C_3 : \quad \sum_{k \in K} a_{k,n} f_{k,n} \leq f_{n,\text{max}}, \forall n \in N,$$

$$C_4 : \quad \sum_{n \in N} a_{k,n} = 1 \forall k \in K,$$

$$C_5 : a_{k,n} \in \{0, 1\}, \forall k \in K, \forall n \in N,$$

where $T_{k,\text{max}}$ is the maximum tolerable latency of the $k$th user, $P_{k,\text{max}}$ is the maximum uplink transmission power of the $k$th user, and $f_{n,\text{max}}$ is the total computation resources in the $n$th server. The constraint $C_1$ guarantees that the service latency experienced by individual users do not exceed their maximum tolerable latency. The constraint $C_2$ restricts the transmission power of each user. The constraint $C_3$ ensures that the computation resources on individual servers are not allocated in excess. The constraints $C_4$ and $C_5$ ensure that an individual user can be served by only one edge server at one time.

With the integer variables in the task assignment $A$ and the highly non-convex objective function $T$, the above optimization problem turns out to be a mixed-integer nonlinear programming problem (MINLP) which is difficult to solve [20].

IV. THE iRAR ALGORITHM

In this section, we develop the joint radio and computation resource management (iRAR) algorithm that efficiently solves problem $\mathcal{P}_0$ as shown in Fig. 1. First, we transform the problem to an equivalent problem based on the uplink/downlink duality. Then, we relax the integer variables in task assignments into continuous variables. We solve the relaxed problem based on the block coordinate descent method. Finally, we convert the continuous task assignments to integer task assignments.

A. Problem Transformation

We transform the uplink power control and receiving beamforming problem into a downlink beamforming problem by leveraging the uplink/downlink duality in MIMO systems [21]. Then, the achievable data rate of the $k$th user in the downlink channel is formulated as

$$R_k = B \log \left( 1 + \frac{\|H^k w_k\|_2^2}{\sum_{j \neq k, j \in K} \|H^k w_j\|_2^2 + \sigma^2} \right), \quad (9)$$

where $w_k = [w^1_k, w^2_k, ..., w^L_k]^T$ and $w^l_k \in \mathbb{C}^{1 \times M}$ is the downlink beamforming vector of the $l$th BS for the $k$th user. Denote $W = \{w_k | k \in K\}$ as the set of downlink beamforming vectors. Then, problem $\mathcal{P}_0$ can be equivalently transformed into

$$\mathcal{P}_1 : \min_{\{A,W,V,F\}} \quad T$$

$$\text{s.t.} \quad C_1, C_3, C_4, C_5,$$

$$C_2 : \sum_{l \in L} \|w^l_k\|_2^2 \leq P_{k,\text{max}}, \forall k \in K. \quad (10)$$

After deriving the downlink beamforming vectors $W$, the uplink receive beamforming is

$$v_k = \frac{w^T_k}{\|w_k\|_2^2}, \forall k \in K, \quad (11)$$

and the uplink transmission power is

$$p_k = \|w_k\|_2^4, \forall k \in K. \quad (12)$$
To solve problem $\mathcal{P}_1$, we relax binary variables $a_{k,n}$ to continuous variables $\tilde{a}_{k,n}$ and denote $\tilde{A} = \{\tilde{a}_{k,n} | k \in \mathcal{K}, n \in \mathcal{N}\}$. The relaxed problem becomes

$$\mathcal{P}_2 : \min_{(\tilde{A}, \mathcal{W}, \mathcal{F})} \quad T$$

s.t. $\quad C_1, C_2, C_3, C_4,$

$$\tilde{C}_5 : \tilde{a}_{k,n} \in [0, 1], \forall k \in \mathcal{K}, \forall n \in \mathcal{N}. \quad (13)$$

Next, we solve problem $\mathcal{P}_2$ based on the block coordinate descent method.

B. Beamforming Design with Fixed $\tilde{A}$ and $\mathcal{F}$

When the task assignment $\tilde{A}$ and computation resource allocation $\mathcal{F}$ are given, problem $\mathcal{P}_2$ is simplified as

$$\mathcal{P}_3 : \max_{\mathcal{W}} \quad -\sum_{k \in \mathcal{K}} \frac{d_k}{\rho_k}$$

s.t. $\quad C_1 : R_k \geq t_k; C_2,$

where $t_k = \frac{d_k}{\left(\frac{r_k}{\sum_{n \in \mathcal{N}} \tilde{a}_{k,n} (\tau_{k,n} + \frac{\rho_k}{\rho_{k,n}})}\right)}$. The constraints $C_3, C_4$ and $\tilde{C}_5$ in problem $\mathcal{P}_2$ do not apply to problem $\mathcal{P}_3$. Solving the problem is challenging because of the non-convexity of the objective function and constraints.

Since the phase of $w_k$ does not change the objective function and constraints of problem $\mathcal{P}_3$, constraint $C_1$ is equivalent to the following second order conic (SOC) constraints $C_1$ [22]:

$$\|r_k\|_2 \leq \sqrt{1 + \frac{1}{(2t_k^2 - 1)} \text{Re}\{R_{kk}\}}, \forall k \in \mathcal{K}, \quad (15)$$

where $r_k = [R_{k1}, R_{k2}, ..., R_{kk}, \sigma]^T$, $R_{kk} = \mathbf{H}^H w_k$, and $\text{Re}\{\cdot\}$ is the real part of a complex variable. In this way, we transform non-convex constraint $C_1$ to convex constraint $\tilde{C}_1$.

The problem $\mathcal{P}_3$ with convex constraints $\tilde{C}_1$ is still hard to be solved because of the highly complicated and non-convex objective function. Since it is proved that the weighted sum rate utility maximization problem is equivalent to the weighted sum mean square error (MSE) minimization problem [23], problem $\mathcal{P}_3$ is transformed into

$$\min_{\{\rho, \mathcal{F}, e\}} \sum_{k \in \mathcal{K}} d_k (\rho_k e_k - \log \rho_k)$$

s.t. $\quad \tilde{C}_1, \tilde{C}_2, \quad (16)$

where $\rho_k$ is the MSE weight for the $k$th user, and $e_k$ is the corresponding MSE defined as

$$e_k = \mathbb{E}[(u_k^H y_k - s_k)^2] = u_k^H \left(\sum_{j \in \mathcal{K}} \left|\mathbf{H}^H w_{kj}\right|^2 + \sigma^2 I\right) u_k$$

$$-2\text{Re}\{u_k^H w_k w_k^H\} + 1,$$

where $u_k$ is defined as the receiver of the $k$th user, $\mathbb{E}[\cdot]$ is the mathematical expectation of a random variable, and $(\cdot)^H$ is the operator of the conjugate transpose. Denote $\rho = \{\rho_k | k \in \mathcal{K}\}$, $e = \{e_k | k \in \mathcal{K}\}$ and $u = \{u_k | k \in \mathcal{K}\}$.

The problem in Eq.(16) is convex with respect to each of individual optimization variables, e.g., $\rho$, $e$ or $u$. Therefore, it can be solved based on the block coordinate descent method and its convergence is guaranteed [23]. The pseudo codes are summarized in Algorithm 1.

At the beginning, the downlink beamforming vectors are initialized. Given $\mathcal{W}$, the MMSE receiver is the optimal receiver [23]. Therefore,

$$u_k = \left(\sum_{j \in \mathcal{K}} \left|\mathbf{H}^H w_{kj}\right|^2 + \sigma^2 I\right)^{-1} \mathbf{H}^H w_k, \forall k \in \mathcal{K}. \quad (18)$$

Then, with the given $\mathcal{W}$ and $u$, the optimal MSE weight $\rho_k$ of the $k$th user can be calculated as

$$\rho_k = e_k^{-1}, \forall k \in \mathcal{K}. \quad (19)$$

Next, with the given $\rho$ and $u$, the problem in Eq.(16) turns out to be a quadratic programming problem

$$\min_{\mathcal{W}} \sum_{k \in \mathcal{K}} w_k^H \left(\sum_{j \in \mathcal{K}} \rho_j d_j \left|\mathbf{H}^H u_j\right|^2\right) w_k$$

$$-2\sum_{k \in \mathcal{K}} \rho_k d_k \text{Re}\{u_k^H w_k w_k^H\}$$

s.t. $\quad \tilde{C}_1, \tilde{C}_2, \quad (20)$

which is a standard convex problem that can be efficiently solved by convex programming solvers, e.g., CVX [20].

C. Task Placement with Fixed $\mathcal{W}$ and $\mathcal{F}$

When the downlink beamforming $\mathcal{W}$ and computation resource allocation $\mathcal{F}$ are given, problem $\mathcal{P}_2$ can be simplified as

$$\mathcal{P}_4 : \min_{(\mathcal{A})} \sum_{k \in \mathcal{K}, n \in \mathcal{N}} \tilde{a}_{k,n} (\tau_{k,n} + \frac{\rho_k}{\rho_{k,n}})$$

s.t. $\quad C_1, C_3, C_4, \tilde{C}_5, \quad (21)$

which is a standard linear programming problem that can be readily solved by linear programing solvers.

D. Computation Resource Allocation with Fixed $\mathcal{W}$ and $\tilde{A}$

When the downlink beamforming $\mathcal{W}$ and task assignment $\tilde{A}$ are given, problem $\mathcal{P}_2$ can be reduced to

$$\mathcal{P}_5 : \min_{\{\mathcal{F}\}} \sum_{k \in \mathcal{K}, n \in \mathcal{N}} \tilde{a}_{k,n} \frac{\rho_k}{\rho_{k,n}}$$

s.t. $\quad C_1, C_3. \quad (22)$

Lemma 1: Problem $\mathcal{P}_5$ is strictly convex with respect to the computation resource allocation $\mathcal{F}$.
Algorithm 2: The iRAR algorithm

Input: $T_{i}^{\text{max}}, T_{j}^{\text{max}}, f_{m,n}, A_{0}, F_{0}$.

1. $\bar{A} \leftarrow A_{0}, F \leftarrow F_{0}$;
2. while True do
   3. $W \leftarrow$ solve problem $P_{3}$ with fixed $\bar{A}$ and $F$;
   4. $\bar{A} \leftarrow$ solve problem $P_{5}$ with fixed $W$ and $F$;
   5. $F \leftarrow$ solve problem $P_{5}$ with fixed $\bar{A}$ and $W$;
   6. if convergence then
      7. break;
8. Determine $V$ and $P$ according to Eq.(11) and Eq.(12);
9. Determine $A$ according to Eq.(24);

<table>
<thead>
<tr>
<th>Table I USER APPLICATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data size (Mbits)</td>
</tr>
<tr>
<td>Type I</td>
</tr>
<tr>
<td>Complexity (GFLOPS)</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>Tolerable latency (s)</td>
</tr>
<tr>
<td>0.2</td>
</tr>
</tbody>
</table>

Proof: For any feasible $f_{m,n}, f_{i,j}, \forall m, i \in K, \forall n, j \in N$,

$$\frac{\partial^{2}T}{\partial f_{i,j} \partial f_{m,n}} = \begin{cases} 2\bar{a}_{i,j}c_{n} \frac{1}{(f_{i,j})^{2}}, & i = j \text{ and } m = n, \\ 0, & i \neq j \text{ or } m \neq n, \end{cases}$$ (23)

The Hessian matrix $T = \frac{\partial^{2}T}{\partial f_{i,j} \partial f_{m,n}}_{K \times K}$ is symmetric and positive definite. The constraints $C_{1}$ and $C_{3}$ are convex with respect to $F$. Hence, $P_{3}$ is strictly convex with respect to $F$.

Based on the above mathematical analysis, we develop the joint radio and computation resource management (iRAR) algorithm which solves problem $P_{2}$ by sequentially fixing two of the variables while updating the other one. The pseudo code of the iRAR algorithm is presented in Algorithm 2.

Theorem 1: The iRAR algorithm converges to the optimal solution of problem $P_{2}$.

Proof: Since the iRAR algorithm is designed based on the block coordinate descent method, the convergence and optimality of the iRAR algorithm can be proved by showing that problem $P_{2}$ is strictly convex with respect to each block of variables, e.g., $\bar{A}, W$ and $F$ [24]. The convexity of problem $P_{2}$ has been proved in the Subsections IV-B, IV-C, and IV-D. Hence, the convergence and optimality of the iRAR algorithm are guaranteed.

Finally, we convert the continuous task assignment $\bar{a}_{k,n}$ to the integer task assignment $a_{k,n}$ according to

$$a_{k,n} = \begin{cases} 1, & n = \arg \max_{j \in N} \bar{a}_{k,j}, \\ 0, & \text{otherwise}. \end{cases}$$ (24)

V. SIMULATION AND ANALYSIS

In this section, we evaluate performance of the iRAR algorithm through network simulations. We consider a mobile edge computing network with 20 users, 20 BSs and 10 servers. BSs and users are randomly distributed in a 500x500 area. The total system bandwidth $B$ is 10 MHz. Each BS is equipped with 4 antennas, and each user has one antenna. The maximum transmit power of an individual user is 20dBm. The path loss between BSs and users is modeled as $140.7 + 36.7 \log_{10}(d)$, where $d$ is the geographical distance in kilometers. The noise power density is $\sigma^{2} = -164$ dBm. The computation capacity of the edge server is 3000 GFLOPS. The network latency $\tau$ is uniformly distributed between 10 and 50 milliseconds.

We consider three types of user applications in the simulations as listed in Table I. Type I applications have stringent latency requirements with medium data sizes and compute-intensity, e.g., augmented reality. Type II applications have high computation requirements with small data sizes and medium latency-sensitivity, e.g., mathematical calculations. Type III applications have large data sizes but with low compute-intensity and high latency tolerance, e.g., IoT data analysis. In the default simulation setting, the percentages of type I, type II, and type III applications are 40%, 30%, and 30%, respectively. To quantify the user’s satisfaction about computing services, we define a dissatisfaction ratio $\gamma$ as

$$\gamma = \frac{|K_{u}|}{|K|},$$

where $K_{u} = \{ k | T_{k} \geq T_{i}^{\text{max}}, \forall k \in K \}$ is the set of unsatisfied users and $|K_{u}|$ is the number of unsatisfied users.

In the simulations, we compare the iRAR algorithm with the following algorithms:

- **The JCC algorithm:** The joint communication and computation (JCC) algorithm reduces the service latency by optimizing the radio and computation resource allocation. The computation tasks are assigned based on the minimum network latency between users and servers.
- **The JTC algorithm:** The joint task assignment and computation (JTC) algorithm reduces the service latency by optimizing the task assignment and computation resource allocation. The radio resource allocation are optimized to maximize the sum rate of users.
- **The Baseline algorithm:** The baseline algorithm allocates radio resource according to Algorithm 1, assigns the users to their nearby servers with the minimum network latency, and equally shares the computation resources among users in the server.

![Figure 2. The performance vs. the number of users.](image-url)
As compared to the baseline algorithm, the iRAR algorithm achieves the lowest latency and dissatisfaction ratio due to the joint optimization of radio and computation resources. The iRAR algorithm achieves the lowest latency and dissatisfaction ratio when the computation capacity is 1000 GFLOPS. When the computation capacity is 2000 GFLOPS or less, only the iRAR algorithm is able to maintain a zero dissatisfaction ratio. In other words, the other algorithms fail to satisfy all users’ latency requirements. Therefore, the joint radio and computation resource management is necessary to provision low latency MEC.

C. The Impact of Computation Capacity

Fig. 4 shows the impact of the edge server’s computation capacity on the average service latency and dissatisfaction ratio. It is shown that the average service latency and dissatisfaction ratio are reduced with the increment of the edge server’s computation capacity. Because the higher computation capacity helps reduce the computation latency. As compared to the other algorithms, the iRAR algorithm achieves the lowest average service latency and smallest dissatisfaction ratio. As compared to the baseline algorithm, the iRAR algorithm reduces 52% of the average service latency and achieves zero dissatisfaction ratio when the computation capacity is 1000 GFLOPS. When the computation capacity is 2000 GFLOPS or less, only the iRAR algorithm is able to maintain a zero unsatisfactory ratio. In other words, the other algorithms fail to satisfy all users’ latency requirements. Therefore, the joint radio and computation resource management is necessary to provision low latency MEC.

Fig. 5 shows the cumulative distribution function of service latency under different algorithms. This figure validates the advantage of the iRAR algorithm in terms of the service latency.

VI. CONCLUSION

In this paper, we have studied the joint radio and computation resource management problem for latency-constrained applications in mobile edge networks. We have formulated the optimization problem which aims to minimize the summation
of users’ average service latency under practical constraints. We have developed the iRAR algorithm to solve the problem and enables low latency mobile edge computing. We have conducted extensive simulations to evaluate the performance of the proposed iRAR algorithm. The simulation results have showed that the iRAR algorithm can effectively reduce the average service latency while meeting all users’ latency requirements.

REFERENCES
