Distributed Probabilistic Offloading in Edge Computing for 6G-Enabled Massive Internet of Things

Zhuofan Liao, *Member, IEEE*, Jingsheng Peng, Jiawei Huang, *Member, IEEE*, Jianxin Wang, *Senior Member, IEEE*, Jin Wang, *Senior Member, IEEE*, Pradip Kumar Sharma, *Member, IEEE*, and Uttam Ghosh, *Senior Member, IEEE*

***Abstract*—Mobile-edge computing (MEC) is expected to provide reliable and low-latency computation offloading for massive Internet of Things (IoT) with the next generation networks, such as the sixth-generation (6G) network. However, the successful implementation of 6G depends on network densification, which brings new offloading challenges for edge computing, one of which is how to make offloading decisions facing densified servers considering both channel interference and queuing, which is an NP-hard problem. This article proposes a distributed-two-stage offloading (DTSO) strategy to give tradeoff solutions. In the first stage, by introducing the queuing theory and considering channel interference, a combinatorial optimization problem is formulated to calculate the offloading probability of each station. In the second stage, the original problem is converted to a nonlinear optimization problem, which is solved by a designed sequential quadratic programming (SQP) algorithm. To make an adjustable tradeoff between the latency and energy requirement among heterogeneous applications, an elasticity parameter is specially designed in DTSO. Simulation results show that compared to the latest works, DTSO can effectively reduce latency and energy consumption and achieve a balance between them based on application preferences.**

***Index Terms*—6G, channel interference, edge computing, Internet of Things (IoT), nonlinear optimization, offloading, queuing theory.**

I. Introduction

WITH the aid of mobile-edge computing (MEC), Internet of Things (IoT) technology is developing rapidly to connect massive terminal devices as a complex intelligent network. Many widely anticipated future services, including eHealth and autonomous vehicles, can access the IoT network, and offload computation tasks to edge servers in proximity [1]. Driven by IoT enabled data-intensive applications, such as virtual reality (VR)-based gaming, there is a need for technological advancements and evolutions for wireless communications beyond the fifth-generation (5G) networks [2]. The sixth-generation network (6G) is expected to provide ultrareliable and fast communication for edge computing where millions of connected devices and applications could operate seamlessly with high data rates and low latency. The joint application research on MEC with 6G for IoT is becoming future communication needs [3].

However, one of the notable features of 6G is the network densification [4] caused by an extended spectrum toward THz, which reaches about 40–50 BSs/km^2 or even denser. Therefore, end-user devices will be covered by multiple servers at the same time and the servers' coverage area will be highly overlapped. Computation offloading is becoming harder for offloading decisions, queuing at the servers, and bandwidth contention. For the smooth combination of 6G and edge computing for massive IoT, it is urgent to study how to make offloading decisions with low latency and energy costs.

Although there are outstanding studies on offloading decision optimization in multiple servers scenarios with 4G and 5G [5], some factors are considered to be negligible to simplify the model, such as the channel interference [6], [7] and the task queuing on the server [8], [9]. In dense 6G networks, channel interference and task queuing are becoming two important factors that should be considered in offloading decisions. Moreover, heterogeneous industrial and civil applications requests have different requirements for latency and energy costs. Real-time applications have high requirements for the delay, while computation-intensive applications can tolerate delay but have high energy consumption. Therefore, the offloading decision strategy should be designed adaptive to different requirements on latency and energy cost.

Considering the above needs and challenges, this article proposes a distributed-two-stage offloading (DTSO) strategy for offloading decisions to reach an adaptive latency and energy cost for end-users in ultradense networks. The DTSO strategy

Manuscript received September 11, 2020; revised October 8, 2020; accepted October 20, 2020. Date of publication October 23, 2020; date of current version March 24, 2021. This work was supported in part by Degree and Postgraduate Education Reform Project of Hunan Province under Grant 2019JGZD057. *(Corresponding author: Jin Wang.)*

Zhuofan Liao, Jingsheng Peng, and Jin Wang are with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China (e-mail: zfliao@csust.edu.cn; jinwang@csust.edu.cn; jasonpeng@stu.csust.edu.cn).

Jiawei Huang and Jianxin Wang are with the School of Information Science and Engineering, Central South University, Changsha 410083, China (e-mail: jiaweihuang@csu.edu.cn; jxwang@csu.edu.cn).

Pradip Kumar Sharma is with the Department of Computing Science, University of Aberdeen, Aberdeen AB24 3FX, U.K. (e-mail: pradip.sharma@abdn.ac.uk).

Uttam Ghosh is with the Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37235 USA (e-mail: ghosh.uttam@ieee.org).

Digital Object Identifier 10.1109/JIOT.2020.3033298

can be implemented in a decentralized way and has good scalability. The main contributions of this work are as follows.

1) We propose a multiuser to multiserver task offloading problem jointly considering both channel interference and bandwidth contention for ultradense networks, which is an NP-hard problem.
2) To solve the joint offloading problem, a DTSO strategy is proposed. In the first stage of DTSO, a combinatorial optimization problem is formulated to minimize offloading latency and energy. In the second stage, the original problem is converted from a complex 0-1 integer nonlinear optimization problem to a nonlinear optimization one. Finally, an optimal solution is given by the sequential quadratic programming (SQP) algorithm.
3) In DTSO, an elasticity parameter is specially designed to make an adjustable tradeoff between the latency and energy requirement among heterogeneous applications.
4) Compared to the latest offloading strategies, DTSO can effectively reduce latency and energy consumption and achieve a balance between them based on application preferences. Compared to extreme strategy, such as all offloading, DTSO can reduce the average delay by 84% and the total energy consumption by 40% in a 50-user network.

The remainder part of this article is organized as follows. Section II reviews related works. The system model is elaborated in Section III. Section IV gives the problem formulation and develops the DTSO strategy. The simulation results are interpreted in Section V, and finally, this article is concluded in Section VI.

II. Related Work

According to optimization objectives, existing work on the offloading decision can be classified into low latency and low energy consumption. The following is a summary of the existing studies for these three optimization goals.

A. Aiming at Reducing Latency

Sthapit *et al.* [10] proposed new solutions for situations when the cloud or fog is not available. By using the alternating direction multiplier method, Wang *et al.* [11] decomposed the NP-hard mixed-integer nonlinear optimization problem into several smaller subproblems, which are completed in parallel across multiple computing units to speed up the calculation. Some works use Lyapunov optimization to get approximate solutions to optimization problems [12]. The studies above are based on nonlinear optimization methods to obtain approximate solutions. Liao *et al.* [13] focused on low latency by k-means and further improved it by online decision making by a Markov decision and proposed a 1-D algorithm to find the optimal data deployment strategy [14]. He *et al.* [15] proposed an auction-based incentive mechanism, where users and mobile devices participate in the system dynamically.

The works above improve the latency performance of mobile devices and ensure a basic energy consumption performance. However, under highly dynamic demand of mobile users, it is hard to flexibly provide the good latency performance when the power of mobile devices is low. Different from these works, the proposed DTSO strategy can dynamically adjust the optimization objectives of latency and energy consumption.

B. Aiming at Reducing Energy Consumption

In many cases, the optimization problem of task offloading is modeled as an integer optimization problem. In order to reduce the complexity of the problem, some suboptimal algorithms were proposed to make the integer optimization problem be solvable in acceptable time [9], [16]. Zhu *et al.* [17] proposed two approximation algorithms for both single and multimobile device scenarios. Yang *et al.* [18] proposed a suboptimal task scheduling strategy to minimize communication resource consumption under latency constraints. To jointly optimize the computation cost and offloading energy, El Haber *et al.* [19] proposed a low complexity algorithm based on the continuous convex approximation method.

Researches [6] and [20] compared online solutions with precomputed offline solutions. The results show that the precomputed offline strategy can provide better performance in the energy consumption of mobile terminals even if the application is not well understood.

Although the above works can effectively reduce the energy consumption of the device, the offloading latency is rarely considered. Existing solutions may be difficult to adapt to the different energy consumption and delay requirements of massive IoT devices. To fill this gap, the DTSO strategy is designed with a tradeoff parameter to dynamically meet different requirements of end users. For example, if users need better latency experience, the tradeoff parameter between latency and energy consumption can be set to more inclined to optimize latency.

C. Joint Optimization of Latency and Energy Consumption

To jointly optimize latency and energy consumption, researchers used iterative algorithms to obtain approximate optimal solutions. Wang and Zhou [21] proposed an iterative improvement algorithm to achieve the purpose of optimization and computational efficiency. To solve a multiobjective optimization problem, an interior-point method was used in [22]. Eshraghi and Liang [23] designed an effective algorithm called the toric algorithm for uncertain computation amount. They also proved that the toric algorithm can always converge to an alternative form of Karush–Kuhn–Tucker (KKT) point of the original problem.

Muñoz *et al.* [24] proposed a framework for joint optimization of wireless resources and computing resources by using the tradeoff between energy consumption and latency. Guo *et al.* [25] proposed a distributed eDors algorithm, which consists of three subalgorithms from respective of computation offloading selection, clock frequency control, and transmission power allocation. Ding *et al.* [26] transformed the task offloading problem into a convex optimization problem and found the optimal solution.

Through the summary of the above work, we find that optimization of delay and energy cannot meet the needs of

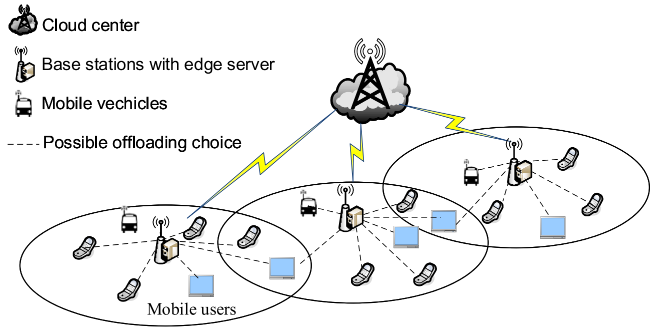


Fig. 1. System model.

edge computing in 6G dense network, and we need to consider more factors. First, channel interference and bandwidth contention will have great impact on latency and energy cost. Then, when the optimization target is latency, we need to ensure that the energy consumption of user equipment does not exceed the given threshold. Finally, if the first consideration is to optimize energy consumption, the quality of user experience also needs to be guaranteed. Based on the above considerations, this work proposed a distributed multiuser and multiserver offloading scheme, which considers an M/M/1 queue on the edge server to reflect queuing and channel conflicts. The energy consumption of user equipment is also formulated not exceeding a given threshold. Moreover, compared with the other solutions, our strategy is designed to run on each edge computing server, which is distributed and has good scalability.

III. System Model

Without loss of generality, we consider the offloading optimization in orthogonal frequency-division multiple access (OFDMA)-based cellular networks [5], [27], which can support multiple access to edge servers. Each base station is equipped with an edge computing server, the former is responsible for transmission and the latter is responsible for computing. Since they are essentially in the same place, we treat the base station and the edge server as a whole, and the transfer time between them is negligible. A single computation task cannot be further segmented, and can be executed locally or on the edge server of the reachable base station. We assumed that each user randomly generates a task within a certain time slot and the task generation of all users obeys Poisson distribution [22], [28].

A. Network Model

As shown in Fig. 1, $B = \{b_1, \dots, b_n\}$ denotes the set of base stations, $U = \{u_1, \dots, u_p\}$ denotes the set of users, and $T = \{t_1, \dots, t_p\}$ represents the set of all computing tasks. Because the user's moving speed is relatively slow, our model does not consider the user's mobility. Obviously, considering the mobility of users will make our model more general, which will be further discussed in our future work. Each base station has a coverage area with radius r_{b_j}, where $\forall b_j \in B$. The coverage status s_{u_i,b_j} of user u_i with base station b_j can be obtained by the relative position relationship between them. If user u_i is covered by the base station b_j, then $s_{u_i,b_j} = 1$, otherwise 0. Because each user will be covered by at least one base station, s_{u_i,b_j} is subject to the following constraint:

$$\sum_{b_j \in B} s_{u_i,b_j} \geq 1 \quad \forall u_i \in U. \tag{1}$$

B. Communication Model

The application scenario of this model is the OFDMA mobile cellular network, where base stations and users devices are connected through wireless channels. High-speed fiber-optic connections are used between base stations and edge servers, and data transmission latency between base stations and servers will be ignored in this model. The data transmission rate between user u_i and base station b_j can be obtained by the Shannon–Hartley theorem

$$r_{u_i,b_j} = \frac{W_{b_j}}{N} \log_2\left(1 + \frac{p_{u_i} g_{u_i,b_j}}{\omega + \sum_{k=1,k\neq i}^{N} p_{u_i} g_{u_i,b_j}}\right) \tag{2}$$

where W_{b_j} denotes the bandwidth of base station b_j, N denotes the number of users of base station b_j service, p_{u_i} denotes the transmission power of user u_i, which is determined by the energy control algorithm between base station and user, g_{u_i,b_j} denotes the channel gain between user u_i and base station b_j, and ω denotes background noise.

Formula (2) fully reflects the channel resource competition and mutual interference between users. It is shown that the more users transmit data, the less bandwidth a single user can share. Meanwhile, the transmission energy of other users in the channel will become the noise of a user. The more users there are, the stronger the noise will be, which will significantly reduce the transmission rate of a user.

The notations of this article are summarized in Table I.

C. Computation Model

For the edge server deployed on base station b_j, we use $\{C_{b_j}, W_{b_j}\}$ to express the CPU frequency and bandwidth. We assume that the amount of result data returned by the base station to the user is very small, so the model does not consider the latency caused by the result returned to the user [5]. For a single user u_i $\forall u_i \in U$, its computing capacity is expressed as C_{u_i}(CPU frequency). For the computing task t_i $\forall t_i \in T$, generated by user u_i, the input data size and the required computing resource amount are represented by a feature binary $\{D_{t_i}, R_{t_i}\}$.

1) Local Computing Model: If task t_i is computed locally, the latency is

$$L_{t_i}^{l} = \frac{R_{t_i}}{C_{u_i}}. \tag{3}$$

Its energy consumption is

$$E_{t_i}^{l} = R_{t_i}\alpha \tag{4}$$

where α represents the energy consumption of each CPU cycle in mobile devices. A widely accepted model, $\alpha = \kappa(C_{u_i})^2$ [5], is adopted here. κ is the energy consumption coefficient.

TABLE I
NOTATIONS

Notation	Definition	Notation	Definition
B	The set of base stations	R_{t_i}	The required computing resource amount of task i
r_{b_j}	Signal coverage radius of b_j	d_{t_i,b_j}	The offloading decision of task i
U	The set of all users	G_j	User group j
T	The set of all computing tasks	T_j	Task group j
s_{u_i,b_j}	If user u_i is covered by the base station b_j, then $s_{u_i,b_j}=1$, otherwise 0	λ_{b_j}	Task arrival rate on base station j
r_{u_i,b_j}	The transmission rate between user i and base station j	μ_{b_j}	Service rate of base station j
W_{b_j}	The bandwidth of base station j	$P_{t_i}^{b_j}$	Probability of user i offload its task to base station j
p_{u_i}	The transmission power of user i	$L_{t_i}^{l}$	Local processing latency of t_i
g_{u_i,b_j}	The channel gain between user i and base station j	$L_{t_i}^{off,b_j}$	Latency of t_i offloading to b_j
ω	Background noise	L^{soj,b_j}	Task processing latency of b_j
C_{b_j}	The CPU frequency of base station j	$E_{t_i}^{l}$	Local processing energy consumption of t_i
D_{t_i}	The input data size of task i	$E_{t_i}^{off,b_j}$	Energy consumption of t_i offloading to b_j
H_T	Total overhead of task set T	β	Trade-off parameter between latency and energy consumption
Δ	Time frame length		

2) MEC Computing Model: A certain offloading scheme can be described as a matrix

$$M_{m\times(n+1)} = \begin{bmatrix} d_{t_1,0}, d_{t_1,b_1} \ldots, d_{t_1,b_n} \\ d_{t_2,0}, d_{t_2,b_1} \ldots, d_{t_2,b_n} \\ \vdots \\ d_{t_m,0}, d_{t_m,b_1} \ldots, d_{t_m,b_n} \end{bmatrix}. \tag{5}$$

Each row in $M_{m\times(n+1)}$ represents an offloading choice for a user task. There are m rows in the matrix and m is the number of users. d_{t_i,b_j} represents the offloading decision of task t_i with the value $\{0, 1\}$. Each row in $M_{m\times(n+1)}$ has $n+1$ elements and n is the number of base stations. If d_{t_i,b_j} takes a value of 1, the task t_i of u_i is executed in the corresponding edge server b_j, and $d_{t_i,0}=1$ is executed locally. In this model, user tasks can only be computed in one place

$$\sum_{d_{t_i,b_j}\in M_i} d_{t_i,b_j} = 1. \tag{6}$$

If the user offloads the task t_i onto the edge server b_j, the latency includes two parts: 1) data transmission latency and 2) task processing latency (including queuing and computing latency). The transmission latency can be expressed as

$$L_{t_i}^{\mathrm{off},b_j} = \frac{D_{t_i}}{r_{t_i,b_j}}. \tag{7}$$

For edge servers on base station b_j, it is assumed that tasks offloaded onto them arrive in Poisson distribution [22], [28]. Considering an M/M/1 queue on b_j, and the main characteristics of the M/M/1 queuing model are as follows: 1) the arrival time of tasks follows Poisson distribution; 2) service time is exponentially distributed; 3) there is only one server; 4) the length of service queue is unlimited; and 5) the number of tasks that can be added to the queue is unlimited. The first "*M*" indicates that the task arrival follows Poisson distribution, the second "*M*" indicates that the task processing time follows an exponential distribution, and the last "1" indicates that there is only one server. The task arrival rate on b_j is

$$\lambda_{b_j} = \frac{\sum_{t_i\in T} d_{t_i,b_j}}{\Delta}. \tag{8}$$

Assuming that the task processing time is exponentially distributed [22], [28], the average task processing time is

$$s_{b_j} = \frac{\sum_{t_i\in T} d_{t_i,b_j} R_{t_i}}{C_{b_j}\sum_{t_i\in T} d_{t_i,b_j}}. \tag{9}$$

Then, the service rate $\mu_{b_j} = (1/s_{b_j})$. So the average processing latency of all tasks offloaded to the base station is

$$L^{soj,b_j} = \frac{1}{\mu_{b_j} - \lambda_{b_j}}. \tag{10}$$

Its energy consumption is

$$E_{t_i}^{\mathrm{off},b_j} = \frac{R_{t_i}}{r_{u_i,b_j}} p_{u_i}. \tag{11}$$

IV. Problem Definition and Solution

In this section, the problem will be formulated and the optimization objective function will be concluded. Then, we will propose our DTSO strategy and introduce the details of the strategy.

A. Problem Definition

According to the previous model, the latency of task t_i can be expressed as follows:

$$L_{t_i} = d_{t_i,0} L_{t_i}^{l} + \sum_{t_i\in T} d_{t_i,b_j}\left(L_{t_i}^{off,b_j} + L^{soj,b_j}\right). \tag{12}$$

Energy consumption can be expressed as

$$E_{t_i} = d_{t_i,0} E_{t_i}^{l} + \sum_{t_i\in T} d_{t_i,b_j}\left(E_{t_i}^{off,b_j} + E^{soj,b_j}\right). \tag{13}$$

An overhead function is introduced to represent the total user overhead. Since the values of delay and energy consumption are in the same order of magnitude, we did not normalize them further

$$H_T = \beta \sum_{t_i \in T} L_{t_i} + (1-\beta) \sum_{t_i \in T} E_{t_i} \tag{14}$$

where β is a parameter between [0, 1], which is used to compromise the user latency and energy consumption. The value of the parameter can vary with the situation. If the emphasis is on energy saving, the value is close to 0, and when the focus is on latency performance, the value is close to 1. Our optimization objective is to minimize the average overhead of all users. Here, we can translate it into the following objective equation:

$$\begin{aligned} &\min \quad \text{object } \min_{M} H_T \\ &\text{s.t.} \begin{cases} C1 : (1) \\ C2 : (6) \\ C3 : d_{t_i,b_j} \in \{0,1\} \quad \forall d_{t_i,b_j} \in M \\ C4 : d_{t_i,0} \in \{0,1\} \quad \forall t_i \in T \\ C5 : \lambda_{b_j} < \mu_{b_j} \qquad \forall b_j \in B. \end{cases} \end{aligned} \tag{15}$$

Constraint $C5$ indicates that the arrival rate of base station tasks cannot exceed the processing rate of the base station. According to queuing theory, when the arrival rate exceeds the processing rate, the processing delay tends to infinity. Therefore, if the arrival rate exceeds the processing rate on a certain base station, a large punitive processing latency will be set. Then, the optimization algorithm will prevent the task from being offloaded to the base station exceeding its processing capacity.

Objective function (15) is a mixed-integer nonlinear programming (MINP) problem and proved to be NP-hard [5], [17]. Inspired by [7], our basic idea is to decompose the original problem into several subproblems, then relax the binary choice in the original problem into offloading probabilities, and finally determine the final offloading decision by the optimal offloading probability.

First, we divide users set U into n groups, and $\{G_1, \ldots, G_n\}$ represents all groups. G_j includes all users within the signal coverage of b_j. Similarly, tasks set T is grouped, and $\{T_1, \ldots, T_n\}$ represents all groups. T_j contains the tasks of all users in G_j. Since a user may be covered by multiple base stations, a single user task may be divided into multiple task groups. For a specific user group G_j, a user has to decide to offload the task to b_j or not. It should be noted that a user may be divided into multiple groups, so a user can offload the task to any base station that covers it. To simplify the problem, we relax binary selection to selection probability. $P_{t_i}^{b_j}$ represents the probability of u_i offload t_i to b_j.

Next, we need to make some adjustments to the previous calculation model. Taking the user group G_j as an example, we use $P^{b_j} = \{P_{t_1}^{b_j}, \ldots, P_{t_{n_j}}^{b_j}\}$ denotes the probability that each user in G_j offload the task to b_j. In this way, we can get the expectation of the latency and energy consumption of each user task. The expectation of local computing latency of task t_i in the group can be expressed as follows:

$$L_{t_i}^{l}\left(P_{t_i}^{b_j}\right) = \left(1 - P_{t_i}^{b_j}\right) \frac{R_{t_i}}{C_{u_i}}. \tag{16}$$

Expectation for local processing energy consumption of task t_i is as follows:

$$E_{t_i}^{l}\left(P_{t_i}^{b_j}\right) = \left(1 - P_{t_i}^{b_j}\right) R_{t_i} \alpha. \tag{17}$$

As before, consider an M/M/1 queue on the edge server. At this time, the expected arrival rate of tasks is

$$\lambda_{b_j}^{*} = \frac{\sum_{t_i \in T_i} P_{t_i}^{b_j}}{\Delta}$$

the average expected processing time of tasks is

$$s_{b_j}^{*} = \frac{\sum_{t_i \in T_i} R_{t_i} P_{t_i}^{b_j}}{C_{b_j} \sum_{t_i \in T_i} P_{t_i}^{b_j}}.$$

Hence, the service rate is

$$\mu_{b_j}^{*} = \frac{1}{s_{b_j}^{*}}.$$

Then, the expectation of task processing latency on base station b_j is

$$L^{soj,b_j}\left(P^{b_j}\right) = \frac{1}{\mu_{b_j}^{*} - \lambda_{b_j}^{*}}. \tag{18}$$

Expected transmission latency of task t_i is

$$L_{t_i}^{off,b_j}\left(P_{t_i}^{b_j}\right) = \frac{P_{t_i}^{b_j} D_{t_i}}{r_{u_i,b_j}}. \tag{19}$$

Expected transmission energy consumption of task t_i is

$$E_{t_i}^{off,b_j}\left(P_{t_i}^{b_j}\right) = \frac{D_{t_i} P_{t_i}^{b_j}}{r_{u_i,b_j}} p_{u_i}. \tag{20}$$

From now on, we can get the expected latency of user task t_i in group T_j as follows:

$$L_{t_i}\left(P_{t_i}^{b_j}\right) = L_{t_i}^{l}\left(P_{t_i}^{b_j}\right) + L_{t_i}^{off,b_j}\left(P_{t_i}^{b_j}\right) + L^{soj,b_j}\left(P^{b_j}\right). \tag{21}$$

The expected energy consumption of user task t_i in group T_j is as follows:

$$E_{t_i}\left(P_{t_i}^{b_j}\right) = E_{t_i}^{l}\left(P_{t_i}^{b_j}\right) + E_{t_i}^{off,b_j}\left(P_{t_i}^{b_j}\right). \tag{22}$$

Therefore, the total expected cost of all user tasks in T_j is

$$H_{T_j}\left(P^{b_j}\right) = \beta \sum_{t_i \in T_j} L_{t_i}\left(P_{t_i}^{b_j}\right) + (1-\beta) \sum_{t_i \in T_j} E_{t_i}\left(P_{t_i}^{b_j}\right). \tag{23}$$

For this user group, our optimization objective function is transformed into the following form:

$$\begin{aligned} &\min \quad \text{object } \min_{P^{b_j}} H_{T_j} \\ &\text{s.t.} \begin{cases} C1 : P_{t_i}^{b_j} \quad \forall t_i \in [0,1] \\ C2 : \lambda_{b_j}^{*} < \mu_{b_j}^{*}. \end{cases} \end{aligned} \tag{24}$$

Considering that (24) is a nonlinear optimization problem, we can use the SQP algorithm [29] to solve it. SQP is one

Algorithm 1: DTSO Strategy

```
Input: Base station set $B$
       Equipment information of $B$
       User set $U$
       Device information of $U$
       Task information of $U$
       Trade-off parameter $\beta$
Output: Offloading decision for all users: $M$
1  $M \leftarrow \emptyset$;
2  Divided users into $n$ groups $G = \{G_1, \dots, G_n\}$;
3  Set $\{P^{b_i}, \dots, P^{b_n}\}$ as the offloading probability;
4  for each $G_j$ in $G$ do
5      $P^{b_j} \leftarrow Solving(24)$;
6  Calculate the average offload probability of each user to the
   reachable location as $\Phi = \{\phi_1, \dots, \phi_p\}$;
7  for each user in $U$ do
8      if random(0,1) $< \phi_{user}$ then
9          base $\leftarrow$ the highest preference base station;
10         $d_{u_{user}, b_{base}} = 1$;
11     else
12         $d_{u_{user}, b_0} = 1$;
13 return $M$;
```

of the most effective methods to solve constrained nonlinear optimization problems. Compared with other algorithms, SQP has many advantages, such as good convergence, high computational efficiency, and strong boundary searching ability. In the iterative process of SQP, one or more quadratic programming (QP) subproblems need to be solved at each step. By solving (24), we can obtain the optimal probability that all users in the user group G_j offload tasks to the server on the base station b_j. In our DTSO strategy, the SQP algorithm is used to calculate the optimal offloading probability. Since the interference between user devices will significantly reduce the data transmission rate, it will lead to an increase of H_{T_j} in (24). Therefore, the SQP algorithm can get a reasonable offloading probability so as to avoid obvious channel interference caused by too many tasks offloading at the same time.

B. DTSO Strategy

In the original problem, the user can choose to process tasks locally or offload part/full of them to servers, which is a 0-1 selectivity optimization problem with NP-hard complexity. In this part, a DTSO strategy is proposed to divide and conquer the original problem. The basic idea of DTSO is as follows.

1) In the first stage, all users are divided into groups according to the base station coverage and then transmit the task information and device information to all reachable base stations. By relaxing the 0-1 selection to the offloading probability, the server on the base station would run the SQP algorithm to obtain the optimal offloading probability of users in every group and then send it back to users.
2) In the second stage, all users calculate the average offloading probability according to the optimal offloading probability to all reachable base stations. Then, the average offloading probability is used to determine whether to offload the task.
3) Specifically, the user will generate a random number between 0 and 1. If the number is less than the average offloading probability, the user task will be offloaded, otherwise, the task will be executed locally. If they decide to offload the task, the user will sort the servers according to their preference for base stations which will be described below. Then, the user selects a base station with the highest preference.

We use $\text{pre}_{u_i}^{b_j}$ to express the preference from user i to BS j, which is determined by the following formula:

$$\text{pre}_{u_i}^{b_j} = \frac{1}{\frac{d_{u_i}^{b_j}}{\psi_{b_j}} + \frac{\text{load}_{b_j}}{n}} \tag{25}$$

where $d_{u_i}^{b_j}$ is distance between user i and BS j. ψ_{b_j} represents the signal coverage of BS j. load_{b_j} indicates the load of BS j, that is, the number of users that have been allocated to it. n is the total number of users.

As shown in Algorithm 1, in the first stage, all users are divided into groups according to the base station coverage and then transmit the task information and device information to all reachable base stations (lines 1–3). By relaxing the 0-1 selection to the offloading probability, the server on the base station would run the SQP algorithm to obtain the optimal offloading probability of users in every group and then send it back to users (lines 4 and 5).

In the second stage, all users can calculate the average offloading probability according to the optimal offloading probability to all reachable base stations (line 6). Then, the average offloading probability is used to determine whether to offload the task (lines 7–13).

Specifically, the user will generate a random number between 0 and 1. If the number is less than the average offloading probability, the user task will be offloaded (lines 8–10), otherwise, the task will be executed locally. If the user task is decided to offload, the user will first query the load of all reachable base stations and then select a base station with the lightest load (lines 11 and 12), that is, the total computing resources required by the task queue on the base station server are the smallest. Finally, the algorithm will output the task offloading decision of all users (line 13).

According to our DTSO, after completing the task grouping, we will use the SQP algorithm to calculate the optimal offloading probability. If too many tasks are assigned to the same base station, SQP will give a relatively low offloading probability. Therefore, when determining the final offloading decision, DTSO will ensure that the tasks offloaded to the base station will not lead to an overload of the base station.

Fig. 2 illustrates a simple example of the DTSO algorithm in the process.

V. Simulation Results

In this section, we evaluate the average task latency and total energy consumption of all users under various conditions, including the impact of user amount, computing capacity, the

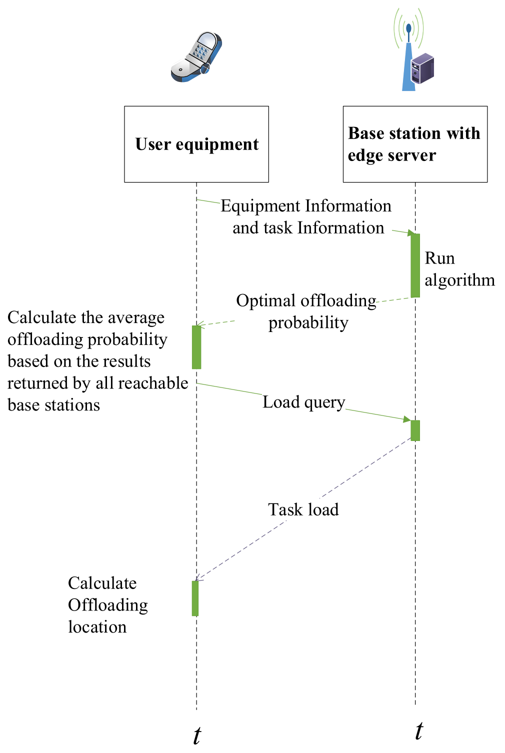

User equipment
Base station with edge server
Equipment Information and task Information
Run algorithm
Optimal offloading probability
Calculate the average offloading probability based on the results returned by all reachable base stations
Load query
Task load
Calculate Offloading location
t
t

Fig. 2. Simple example of the DTSO algorithm.

transmission power of user equipment, and the influence of tradeoff parameter β. All local processing (ALP) schemes, all offloading processing (AOP) scheme, PROMOT [30], and HTOS [5] are used as benchmarks to compare with our proposed DTSO strategy. Because our algorithm is distributed and each subproblem is run by a powerful edge server, we ignore the running time of the algorithm on the server. All the benchmarks are described as follows.

1) ALP, an all-local processing strategy, which means all user tasks are processed directly on the user device. And this extreme strategy can be used as the optimal performance of energy consumption.
2) AOP, an AOP strategy, which means all user tasks are offloaded to the MEC server, and then randomly select one of the reachable base stations to process user tasks.
3) PROMOT [30], a distributed priority offloading mechanism with joint offloading proportion and transmission energy algorithm based on Genetic Algorithm. PROMOT uses an offloading probability to determine if a task should be processed locally or remotely. The probability is computed from the prior information of past offloading feedbacks.
4) HTOS [5], a heuristic algorithm for task offloading scheduling that can find a local optimum.

The experiment is set up in a 400 m × 400 m square area with four base stations, as shown in Fig. 3. The coverage of each base station is set to 150 m, the bandwidth of the base station to 10 MHz, and the CPU frequency of the edge server connected by the base station to 10 GHz. User devices are distributed uniformly at random over the BS coverage regions, and the size of input data of user tasks is randomly within

TABLE II
SIMULATION PARAMETER SETTING

Parameter	**Setting**
Simulation area size	400m×400m [7]
Number of BS	4 [7]
BS signal coverag	150m [7]
BS bandwidth	10 MHz [5]
CPU frequency of MEC server	10 GHz [5]
CPU frequency of mobile device	1 GHz [5]
Data size of task	[600,1000] KB [5]
CPU cycles required by task	[400,600] Megacycles [5]
Transmission power of user device	0.5W [31]
Energy consumption parameter	5×10^{-27} [31]
Channel gain model	L[dB] = 140.7 + 36.7log10d[km] [32]
Time slot	5S
Trade off parameter	0.5

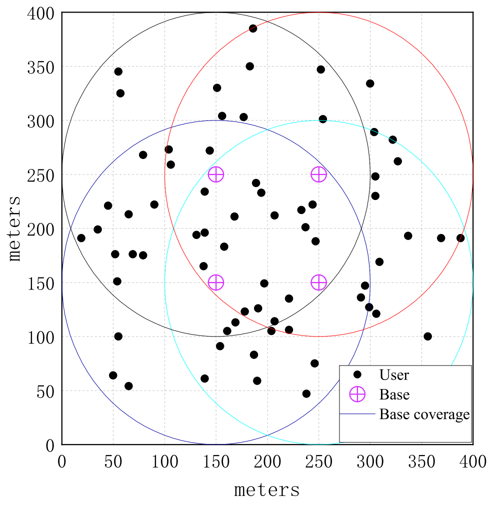


Fig. 3. System setup.

[600,1000] KB. The CPU cycles required by task are randomly within [400,600] Megacycles. The transmission power of the user device is set to 0.5 W [31], the CPU frequency to 1 GHz, and the energy consumption parameter to 5×10^{-27} [18]. The background noise is set to −100 dBm [5]. The channel gain model between base station and user device is set to $L[\text{dB}] = 140.7 + 36.7\log10d[\text{km}]$ [32]. The time slot Δ is set to 5 s and the value of β is to 0.5. Table II lists the parameters used in the simulations.

A. Impact of User Amount

From Fig. 4, we can conclude that DTSO performs best in terms of overhead which is determined by the user latency and energy consumption. When the user amount is less than 15, the overhead of AOP is the smallest, while when the user amount is more than 15, the overhead of AOP increases sharply. This is because when many users offloading together, there are significant channel interference and bandwidth contention. At the

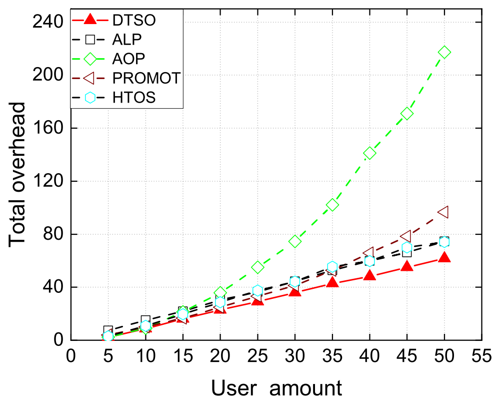


Fig. 4. Total overhead under different user amount.

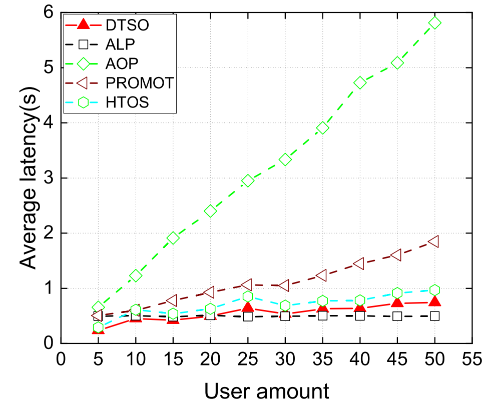


Fig. 5. Average latency under different user amount.

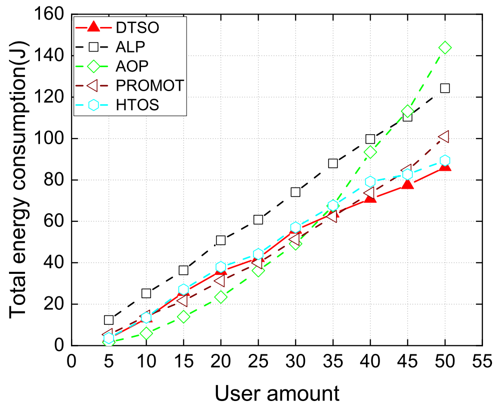


Fig. 6. Total energy consumption under different user amount.

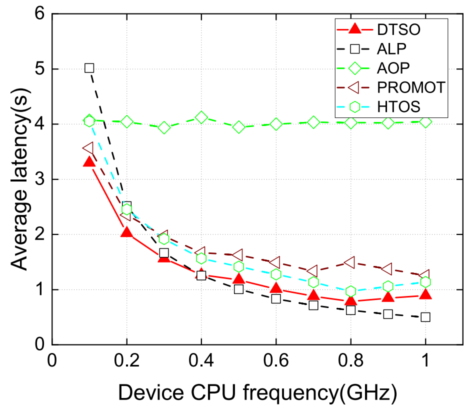


Fig. 7. Impact of device frequency on average latency.

same time, the processing time of tasks on the server increases because of the extension of queuing time.

Fig. 5 shows the average task latency performance of various offloading schemes. AOP gets the highest average latency, which is due to the interference of transmission caused by large numbers of users offloading together, which leads to the reduction of data transmission rate, and the queuing situation on the server also increases the processing time of tasks. When the number of the task is less than 20, the latency performance of DTSO is better than that of ALP, because the data transmission rate is less disturbed and the processing time on the server is shorter. The latency performance of HTOS is slightly worse than that of DTSO, while PROMOT is the worst except AOP. Adjusting the value of tradeoff parameter β can make DTSO outperform ALP in latency. The tradeoff parameter is used to make a tradeoff between latency and energy consumption optimization. If β is set to 1, which means only the latency performance is optimized, then DTSO will choose to offload a small part of tasks to the edge server. When the number of tasks transmitted in the channel is relatively small, the signal interference between user equipment will not significantly reduce the data transmission rate. The processing latency of the edge server is significantly less than that of the local device. Therefore, setting the tradeoff parameter close to 1 can make DTSO's latency performance outperform ALP.

Fig. 6 presents the total energy consumption of user devices with various offloading schemes. When the user amount is less than 35, the total energy consumption of AOP is the lowest, and DTSO is similar to HTSO and PROMOT. When the user amount increases, DTSO achieves the best energy performance. With the increase in user amount, the energy consumption of AOP increases sharply, which is mainly due to the significant increase in transmission energy consumption caused by the decrease in the data transmission rate.

B. Impact of Device Frequency

Besides, we also analyzed the impact of the device frequency. Figs. 7 and 8 show the impact of different device frequency on the average latency and total energy consumption of various offloading strategies in the case of 35 users and the transmission power of each user device is set to 0.5 W.

The information we can get from Fig. 7 is that with the increase of the frequency of the device, all schemes except the AOP get lower task processing latency. ALP has the greatest latency reduction, and other strategies can also achieve good rewards. That is, because the increase in computing capacity of the device has greatly increased the available resources of all computing tasks. At the same time, we notice that the growth rate of rewards is decreasing with the increase in CPU frequency. This will be further explored in our future work.

Fig. 8 shows that the total energy consumption of all strategies except AOP will increase when the frequency of the device is increased. Because using extra computing resources requires extra energy. While the total energy consumption

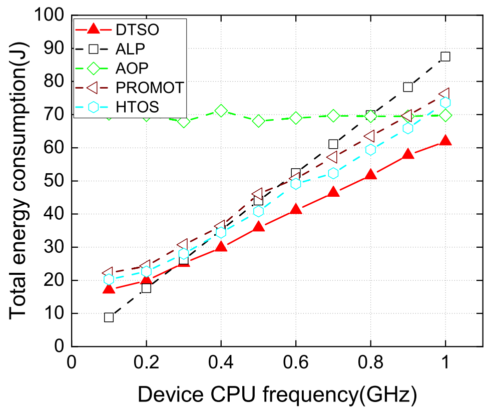


Fig. 8. Impact of device frequency on total energy consumption.

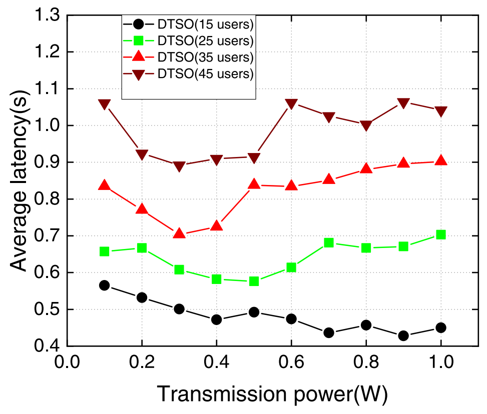


Fig. 9. Impact of device transmission power on average latency.

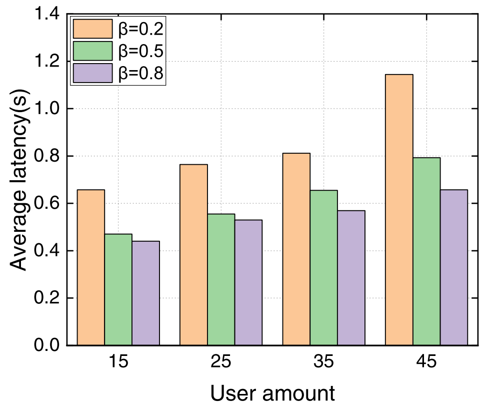


Fig. 10. Influence of β on average latency of DTSO algorithm.

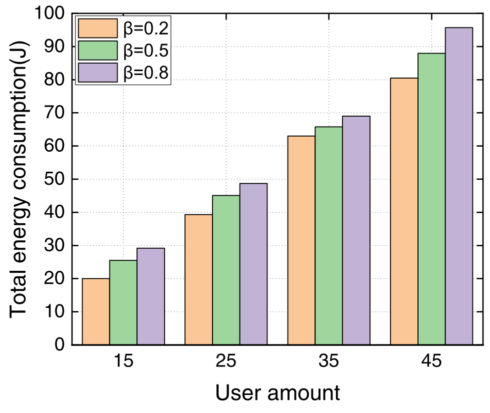


Fig. 11. Influence of β on total energy consumption of DTSO algorithm.

of DTSO al is the smallest among all offloading strategies. That is, because DTSO can make a good tradeoff between latency and energy consumption, rather than only focusing on the reduction of latency but ignoring the increase in energy consumption.

Fig. 9 demonstrates the impact of transmission power on the average latency of DTSO. At this point, the main frequency of all devices is set to 1 GHz. When the number of users is 25, 35, and 45, with the increase of transmission power, the average latency first decreases and then rebounds. When the number of users is 15, with the increase of transmission power, the average latency decreases, but when the transmission power increases to a certain value (0.7 W), the decrease of latency becomes less obvious. The main reason is that the increase of transmission power first increases the data transmission rate, and then decreases the transmission rate due to the increase of mutual interference between devices.

C. *Influence of Tradeoff Parameter β*

We further explored the influence of the tradeoff parameter β on the proposed DTSO. We tested the average latency and total energy consumption of all users when β was set to 0.2, 0.5, and 0.8 for 15, 25, 35, and 45 users, respectively.

As can be seen from Fig. 10, when β is larger, the average latency of users is smaller. That is, because the larger the value of β, the better the algorithm focuses on the optimization of latency performance. This is also in line with our interpretation of β in (14).

Similarly, Fig. 11 shows that when β is larger, the algorithm is more focused on optimizing users' energy consumption.

As a summary of this part, we can effectively control the optimization focus of the algorithm by weighing the parameter β. Users can adjust β according to actual needs to get a better experience. If the user's power is sufficient, then he can set β a little larger to reduce the device latency. On the contrary, set β a little bit smaller to get a longer endurance.

From the previous simulation results, We can conclude that DTSO can make a good compromise between user latency and energy consumption to ensure that both latency and energy consumption is at a low level. Especially, DTSO can achieve the best performance when the number of users is large. Compared with AOP, proposed DTSO can reduce the average latency by 86% and the total energy consumption by 40% in a 50-users network.

VI. Conclusion

Considering both channel interference and bandwidth contention, this work studied a distributed offloading decision strategy for ultradense networks, such as 6G, by grouping users according to the coverage area of the base station and relaxing the decision making of task offloading with probability. Simulation results show that compared to the latest existing works, our strategy can not only reach low latency and energy but also make a good tradeoff between latency and energy cost.

References

[1] J. Yuan, Q. He, M. Matthaiou, T. Q. S. Quek, and S. Jin, "Toward massive connectivity for IoT in mixed-ADC distributed massive MIMO," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1841–1856, Mar. 2020.

[2] W. Dong, Z. Xu, X. Li, and S. Xiao, "Low-cost subarrayed sensor array design strategy for iot and future 6G applications," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4816–4826, Jun. 2020.

[3] M. Latva-Aho and K. Leppänen, "Key drivers and research challenges for 6G ubiquitous wireless intelligence," 6G Flagship Univ. Oulu, Oulu, Finland, White Paper, 2019.

[4] K. David, J. Elmirghani, H. Haas, and X.-H. You, "Defining 6G: Challenges and opportunities [from the guest editors]," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 14–16, Sep. 2019.

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

[6] M. Kamoun, W. Labidi, and M. Sarkiss, "Joint resource allocation and offloading strategies in cloud enabled cellular networks," in *Proc. IEEE Int. Conf. Commun.*, London, U.K., Sep. 2015, pp. 5529–5534.

[7] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Paris, France, Apr. 2019, pp. 10–18.

[8] H. Zhang, J. Guo, L. Yang, X. Li, and H. Ji, "Computation offloading considering fronthaul and backhaul in small-cell networks integrated with MEC," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Atlanta, GA, USA, 2017, pp. 115–120.

[9] C. You and K. Huang, "Multiuser resource allocation for mobile-edge computation offloading," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, 2016, pp. 1–6.

[10] S. Sthapit, J. Thompson, N. M. Robertson, and J. R. Hopgood, "Computational load balancing on the edge in absence of cloud and fog," *IEEE Trans. Mobile Comput.*, vol. 18, no. 7, pp. 1499–1512, Jul. 2019.

[11] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An ADMM framework," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2763–2776, Mar. 2019.

[12] C. F. Liu, M. Bennis, M. Debbah, and H. Vincent Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019.

[13] Z. Liao, R. Zhang, S. He, D. Zeng, J. Wang, and H.-J. Kim, "Deep learning-based data storage for low latency in data center networks," *IEEE Access*, vol. 7, pp. 26411–26417, 2019.

[14] Z. Liao, J. Peng, Y. Chen, J. Zhang, and J. Wang, "A fast *Q*-learning based data storage optimization for low latency in data center networks," *IEEE Access*, vol. 8, pp. 90630–90639, 2020.

[15] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A truthful online mechanism for collaborative computation offloading in mobile edge computing," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4832–4841, Jul. 2020.

[16] 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.

[17] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4854–4866, Jun. 2019.

[18] X. Yang *et al.*, "Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff," *IEEE Access*, vol. 6, pp. 16665–16677, 2018.

[19] E. El Haber, T. M. Nguyen, and C. Assi, "Joint optimization of computational cost and devices energy for task offloading in multi-tier edge-clouds," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3407–3421, May 2019.

[20] W. Labidi, M. Sarkiss, and M. Kamoun, "Energy-optimal resource scheduling and computation offloading in small cell networks," in *Proc. IEEE 11th Int. Conf. Wireless Mobile Comput. Netw. Commun. (WiMob)*, Sydney, NSW, Australia, 2015, pp. 794–801.

[21] W. Wang and W. Zhou, "Computational offloading with delay and capacity constraints in mobile edge," in *Proc. IEEE Int. Conf. Commun.*, Paris, France, 2017, pp. 1–6.

[22] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. S. Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, early access, May 12, 2020, doi: 10.1109/TMC.2020.2994232.

[23] N. Eshraghi and B. Liang, "Joint offloading decision and resource allocation with uncertain task computing requirement," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Paris, France, Apr. 2019, pp. 1414–1422.

[24] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[25] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 2, pp. 319–333, Feb. 2019.

[26] Y. Ding, C. Liu, X. Zhou, Z. Liu, and Z. Tang, "A code-oriented partitioning computation offloading strategy for multiple users and multiple mobile edge computing servers," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4800–4810, Jul. 2020.

[27] Y. Choi, S. Park, and S. Bahk, "Multichannel random access in OFDMA wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 603–613, Mar. 2006.

[28] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE INFOCOM Conf. Comput. Comput.*, Honolulu, HI, USA, Apr. 2018, pp. 207–215.

[29] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 1999.

[30] J. Wang *et al.*, "A probability preferred priori offloading mechanism in mobile edge computing," *IEEE Access*, vol. 8, pp. 39758–39767, 2020.

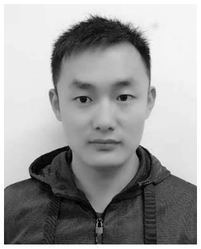
[31] Y. Yang, Y. Ma, W. Xiang, X. Gu, and H. Zhao, "Joint optimization of energy consumption and packet scheduling for mobile edge computing in cyber-physical networks," *IEEE Access*, vol. 6, pp. 15576–15586, 2018.

[32] X. Chu, D. Lopez-Perez, Y. Yang, and F. Gunnarsson, *Heterogeneous Cellular Networks Theory, Simulation and Deployment*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

Zhuofan Liao (Member, IEEE) received the Ph.D. degree in computer science from Central South University, Changsha, China, in 2012.

From 2017 to 2018, she worked as a visiting scholar with the University of Victoria, Victoria, BC, Canada, supported by China Scholarship Council. She is currently an Assistant Professor with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha. Her research interests includes wireless networks optimization, big data, and edge computing for 5G. She has published papers in leading transaction and conferences as the first authored in the above areas.

Dr. Liao has served as a reviewer of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.



Jingsheng Peng is currently pursuing the master's degree with Changsha University of Science and Technology, Changsha, China.

He is good at C/C++ programming under Linux, and also familiar with Python. His research interests include mobile-edge computing and big data placement optimization.

Jiawei Huang (Member, IEEE) received the bachelor's degree from the School of Computer Science, Hunan University, Changsha, China, in 1999, and the master's and Ph.D. degrees from the School of Computer Science and Engineering, Central South University, Changsha, in 2004 and 2008, respectively.

He is currently a Professor with the School of Computer Science and Engineering, Central South University. His research interests include performance modeling, analysis, and optimization for wireless networks, and data center networks.

Jianxin Wang (Senior Member, IEEE) received the bachelor's and master's degrees in computer engineering and the Ph.D. degree in computer science from Central South University, Changsha, China, in 1992, 1996, and 2001, respectively.

He is currently the Chair and a Professor with the Department of Computer Science, Central South University. His current research interests include algorithm analysis and optimization, parameterized algorithm, bioinformatics, and computer network.

Jin Wang (Senior Member, IEEE) received the M.S. degree from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2005, and the Ph.D. degree from Kyung Hee University, Seoul, South Korea, in 2010.

He is currently a Professor with Changsha University of Science and Technology, Changsha, China. He has published more than 300 international journal and conference papers. His research interests mainly include wireless *ad hoc* and sensor network, network performance analysis, and optimization.

Prof. Wang is a a Member of ACM.

Pradip Kumar Sharma (Member, IEEE) received the Ph.D. degree in computer science and engineering (CSE) from Seoul National University of Science and Technology, Seoul, South Korea, in August 2019.

He worked as a Postdoctoral Research Fellow with the Department of Multimedia Engineering, Dongguk University, Seoul. He was an Software Engineer with MAQ Software, Hyderabad, India, and involved in variety of projects, and proficient in building largescale complex data warehouses, OLAP models, and reporting solutions that meet business objectives and align IT with business. He is an Assistant Professor of Cybersecurity with the Department of Computing Science, University of Aberdeen, Aberdeen, U.K. He has published many technical research articles in leading journals of the IEEE, Elsevier, Springer, and MDPI. Some of his research findings are published in the most cited journals. His current research interests include cybersecurity, blockchain, edge computing, SDN, SNS, and the IoT security.

Dr. Sharma received the Top 1% Reviewer of computer science by Publons Peer Review Awards in 2018 and 2019, and Clarivate Analytics. He has also been invited to serve as the Technical Programme Committee Member and the Chair of several reputed international conferences, such as the IEEE ICC 2019, the IEEE MENACOMM 2019, and 3ICT 2019. He has been an Expert Reviewer of the IEEE Transactions, Elsevier, Springer, and MDPI journals and magazines. He has been serving as a Guest Editor for international journals of certain publishers, such as Springer, MDPI, and JIPS. He is currently an Associate Editor of *Human-Centric Computing and Information Sciences* and the *Journal of Information Processing Systems*.

Uttam Ghosh (Senior Member, IEEE) received the Ph.D. degree in electronics and electrical engineering from IIT Kharagpur, Kharagpur, India, in 2013.

He has a Postdoctoral Experience with the University of Illinois at Urbana–Champaign, Champaign, IL, USA, Fordham University, New York, NY, USA, and Tennessee State University, Nashville, TN, USA. He is working as an Assistant Professor of Practice with the Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville. He has published 50 articles in reputed international journals, including the IEEE Transactions, Elsevier, Springer, IET, Wiley, InderScience, and IETE, and in top international conferences sponsored by the IEEE, ACM, and Springer. His main research interests include cybersecurity, computer networks, wireless networks, information-centric networking, and software-defined networking.

Dr. Ghosh has served as a Technical Program Committee Member of renowned international conferences, including ACM SIGCSE, the IEEE LCN, IEMCON, STPSA, SCS SpringSim, and the IEEE COMPSAC. He is a Member of AAAS, ASEE, ACM, and Sigma Xi. He has conducted several sessions and workshops related to cyber–physical systems (CPS), SDN, the IoT, and smart cities, as the Co-Chair of top international conferences, including the IEEE GLOBECOM, MASS, SECON 2019, CPSCOM 2019, IEMCON 2019, and ICDCS 2017. He is actively working on editing two edited volumes on Emerging CPS, Security, Machine/Machine Learning (CRC Press), a Chapman Hall Big Data Series. He is contributing as a Guest Editor for special issues of the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, *ACM Transactions on Internet Technology*, *Multimedia Tools and Applications* (Springer), and *Internet Technology Letters* (Wiley).