# SCMA-Enabled Multi-Cell Edge Computing Networks: Design and Optimization

Pengtao Liu, Kang An, Jing Lei, Wei Liu, Yifu Sun, Gan Zheng, Fellow, IEEE and Symeon Chatzinotas, Fellow, IEEE

***Abstract*—Multi-access edge computing (MEC) is regarded as a promising approach for providing resource-constrained mobile devices with computing resources through task offloading. Sparse code multiple access (SCMA) is a code-domain non-orthogonal multiple access (NOMA) scheme that can meet the demands of multi-cell MEC networks for high data transmission rates and massive connections. In this paper, we propose an optimization framework for SCMA-enabled multi-cell MEC networks. The joint resource allocation and computation offloading problem is formulated to minimize the system cost, which is defined as the weighted energy cost and latency. Due to the nonconvexity of the proposed optimization problem induced by the coupled optimization variables, we first propose an algorithm based on the block coordinate descent (BCD) method to iteratively optimize the transmit power and edge computing resources allocation by deriving closed-form solutions, and further develop an improved low-complexity simulated annealing (SA) algorithm to solve the computation offloading and multi-cell SCMA codebook allocation problem. To solve the problem of partial state observation and timely decision-making in long-term optimization environment, we put forward a multiagent deep deterministic policy gradient (MADDPG) algorithm with centralized training and distributed execution. Furthermore, we extend the framework to the partial offloading case and propose an algorithm based on alternating convex search for solving the task offloading ratio. Numerical results show that the proposed multi-cell SCMA-MEC scheme achieves lower energy consumption and system latency in comparison to the orthogonal frequency division multiple access (OFDMA) and power-domain (PD) NOMA techniques.**

***Index Terms*—Internet of Things, Sparse Code Multiple Access (SCMA), Multi-Access Edge Computing (MEC), binary offloading, partial offloading, resource management.**

## I. Introduction

Driven by the rapid development of the Internet of Things (IoT), a large number of computation-intensive applications are emerging, such as virtual/augmented reality (VR/AR), self-driving cars, and smart homes [1]. However, the limited computation capability of IoT devices remains a barrier to completing latency-critical tasks. Against this shortcoming, multi-access edge computing (MEC) can provide network-edge computing resources in base stations for resource-constrained users and offer lower latency and energy consumption for IoT devices to perform computation-intensive tasks through task offloading [2], [3]. The research on task offloading can be divided into three aspects: task offloading decision, computation resources allocation, and mobility management [4]. Shu et al. [5] proposed a fine-grained offloading strategy to minimize the task completion time by considering dependencies between subtasks and competition among multiple edge users. In [6], dynamic voltage frequency scaling (DVFS) technology was introduced into the optimization of local computation cost. By optimizing the computational speed, transmit power, and offloading ratio of IoT devices, energy consumption and delay can be significantly reduced. The authors in [7] analyzed the joint optimization of offloading decisions and radio allocation in sliced multi-cell MEC networks. These two subproblems were solved iteratively by an alternate optimization method. The work in [8] considered MEC networks with multiple servers, where a joint task offloading and resource allocation algorithm was proposed to improve the devices' offloading benefits.

Non-orthogonal multiple access (NOMA) schemes allow plenty of devices to share the same communication resources at the same time, including time slots and frequency spectrum. It brings many advantages, such as a large number of connections, higher spectral efficiency, and lower latency [9]. Sparse code multiple access (SCMA) is a NOMA designed in the code domain, which combines multi-dimensional modulation technique with low density spread spectrum [10]. Compared to power-domain (PD) NOMA schemes, SCMA offers the benefits of coding gain and shaping gain, resulting in improved throughput and bit-error-rate (BER) [11]. Simulations demonstrated that SCMA-based systems can achieve higher throughput than PD-NOMA at the cost of more complex detections [12].

Recently, the application of NOMA into MEC for IoT scenarios has been regarded as a promising approach to provide efficient transmission and timely computation for massive mobile devices [13]–[20]. Particularly, multiple devices can simultaneously offload computation tasks to MEC servers through NOMA, further increasing the flexibility and efficiency in computation offloading. More specifically, it is shown in [13] that NOMA plays an important role in reduc-



This work is supported in part by the National Natural Science Foundation of China (No. 61702536), in part by R-STR-5010-00-Z SIGCOM RG of Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg. (Corresponding author: Jing Lei)

P. Liu, J. Lei and W. Liu are with the College of Electronic Science and Technology, National University of Defense Technology, Changsha, China. (e-mail:{liupengtao15, leijing, wliu_nudt}@nudt.edu.cn)

K. An and Y. Sun are with the Sixty-Third Research Institute, National University of Defense Technology, Nanjing, China (e-mail:{ankang89, sunyifu.nudt}@nudt.edu.cn).

G. Zheng is with the Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Loughborough LE113TU, U.K. (e-mail: g.zheng@lboro.ac.uk)

S. Chatzinotas is with Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg, L-1855, Luxembourg (e-mail: symeon.chatzinotas@uni.lu)

ing the energy consumption and latency of task offloading. The latency minimization problem of NOMA assisted MEC was investigated in [14], where authors proposed an efficient layered algorithm to find the optimal offloading strategy for devices with the minimum task execution delay. The work in [15] adopted a partial offloading scheme in the hybrid NOMA enabled MEC network, which solved the problem of minimizing energy consumption under delay restriction through power allocation, time slot scheduling, and offloading strategy. The authors in [16] proposed a NOMA scheme for edge computing awareness and designed a heuristic device clustering and resource allocation algorithm to minimize the energy consumption of MEC users. In [17], the authors exploited NOMA to maximize the energy efficiency in multi-cell MEC networks through a joint radio and computation resources allocation scheme. An SCMA-enabled MEC scheme in IoT scenarios was designed in [18], where the authors emphasize the maximization of sum rate. In [19], a joint resource allocation and task offloading optimization design in a single-cell MEC network enabled by SCMA was proposed to achieve the tradeoff between task latency and energy expenditure.

While most of the aforementioned studies have focused on the single-cell MEC network scenarios, multi-cell SCMA-based MEC systems have not yet been studied. There are several key challenges that need to be considered and addressed. First of all, in the case of multiple cells, multiple terminals can access the BS randomly by sharing the same SCMA codebook at the same time. Therefore, it is challenging to allocate SCMA codebooks to reduce inter-cell interference between devices and improve transmission rates. Secondly, considering the joint optimization of task latency and energy efficiency, we need to properly allocate local computing resources, transmit power for task offloading, and edge computing resources. Additionally, offloading too many tasks reduces the offloading benefits due to interference and competition for limited computing resources, and thus an intelligent offloading strategy is necessary. Finally, for a terminal, it needs to decide not only whether the computation task should be offloaded, but also which SCMA codebook to occupy and which MEC server to offload the task to. The coupling of SCMA codebook allocation and offloading decisions makes the problem even more challenging. Motivated by the above challenges, the contributions of this paper are presented below.

1) We design an optimization framework for SCMA-enabled multi-cell edge computing networks, where the problem of inter-cell interference and SCMA codebook reuse are analyzed. Specifically, a system cost is first formulated to measure the devices' energy expenditure and task execution latency. Under the constraints of the maximum latency of tasks, the limited communication, and computation resources, an initial problem of minimizing the system cost is formulated.

2) Since the formulated problem is non-convex due to the coupled variables and intractable constraints, we solve it by subdividing it into two manageable sub-problems, i.e., resource allocation and task offloading policy. As such, by using the variable substitution method, the non-convex optimization problem of resource allocation is transformed into a convex one, and the closed-form solutions under fixed power and frequency are found respectively. We further propose a joint power and computing resource allocation algorithm based on BCD to obtain the near-optimal power and frequency allocation. Furthermore, the task offloading and multi-cell SCMA codebook allocation (TOCA) based on improved simulated annealing (SA) is proposed with low complexity.

3) In a dynamic environment, where channel and task states are constantly changing, it is difficult to obtain global information. In this case, the proposed SA algorithm may not be applicable. We propose a multiagent deep deterministic policy gradient (MADDPG)-based TOCA algorithm with centralized training and distributed execution to solve the problem of partial state observation and timely decision making.

4) We show that the system cost minimization problem addressed in this paper is a unified framework of energy optimization and latency minimization, which can be achieved by changing the weighted factor. In addition, we extend to the partial offloading case and propose a resource allocation algorithm and partial offloading policy based on alternating convex search.

5) Numerical simulations illustrate that the proposed joint optimization algorithm can minimize the system cost (i.e. the weighted value of energy consumption and time delay), and show that the proposed multi-cell SCMA-MEC framework has significant advantages over OFDMA-MEC and PD-NOMA enabled MEC.

The rest of the paper is organized as follows. Section II designs the model of SCMA-enabled multi-cell edge computing networks and formulates the initial optimization problem. Section III proposes the joint resource allocation and task offloading scheme. Simulation results are presented in Section IV, and conclusions are made in Section V.

## II. System Model

The model of multi-cell SCMA-enabled MEC networks is shown in Fig. 1, where each base station (BS) is equipped with a MEC server providing task offloading services. We denote the set of IoT devices and BSs (MEC servers) in the network as $\mathcal{U} = \{1, 2, \ldots, U\}$ and $\mathcal{N} = \{1, 2, \ldots, N\}$, respectively. Multiple IoT devices can simultaneously offload computing tasks to MEC servers by SCMA. The system model consists of three parts: computing task model, task offloading, and MEC execution model. For easy reading, the key symbols are summarized in TABLE I.

### A. Computing Task Model

We assume that each IoT device $u$ has one computation task at a time, represented as $T_u$. The device can choose to perform the computing task locally or offload it to a nearby MEC server, i.e., binary offloading is adopted in the model [8]. The task $T_u$ can be represented by three-tuple parameters, $(d_u, c_u, t_u^{\max})$, where $d_u$ (in bits) specifies the amount of data for task description, $c_u$ (in cycles) represents the number of CPU calculations for $T_u$, and $t_u^{\max}$ (in seconds) indicates the required completion time of the task. The local computing frequency of user $u$ is defined as $f_u^l$ (in cycles/s). When the

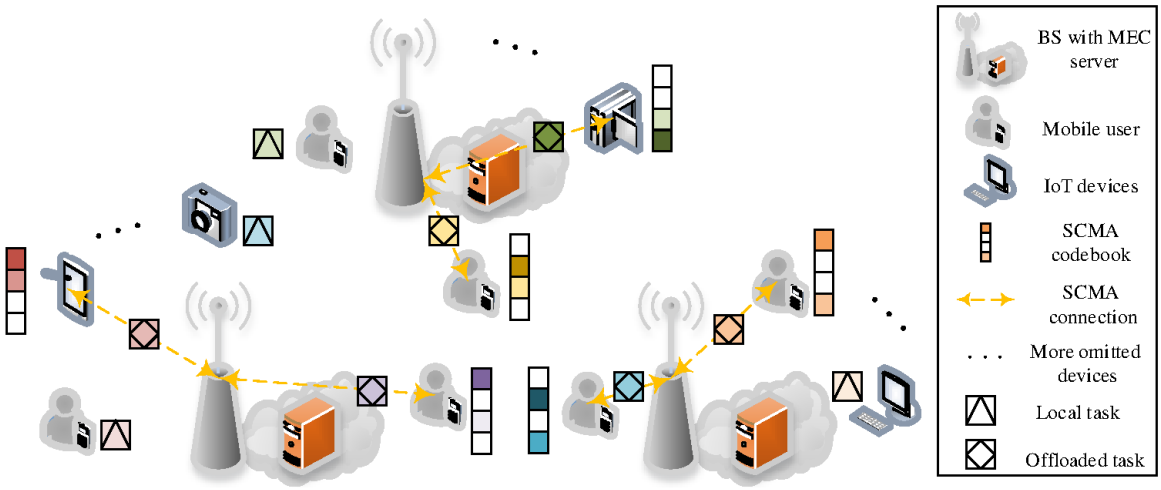



Fig. 1. A diagram of the system model, where multiple devices offload tasks to MEC servers via SCMA.

TABLE I
SUMMARY OF KEY SYMBOLS.

| Symbols | Descriptions |
|---|---|
| $u$ | Index of IoT devices |
| $n$ | Index of BSs (MEC servers) |
| $c$ | Index of SCMA codebooks |
| $k$ | Index of subcarriers |
| $\mathcal{U}$ | Set of IoT devices |
| $\mathcal{N}$ | Set of BSs (MEC servers) |
| $\mathcal{C}$ | Set of SCMA codebooks |
| $\mathcal{K}$ | Set of subcarriers |
| $\mathcal{F}^l$ | Local computation adjustment policy |
| $\mathcal{F}$ | Edge computation resource allocation strategy |
| $\mathcal{S}$ | Computation offloading strategy incorporating multi-cell SCMA codebook allocation |
| $\mathcal{P}$ | Power allocation policy of IoT devices |
| $T_u$ | Computing task of device $u$ |
| $d_u$ | The amount of data for $T_u$ description |
| $c_u$ | The number of CPU calculations for $T_u$ |
| $t_u^{\max}$ | The required completion time of $T_u$ |
| $S_{u,c}^n$ | Relationship indicator of device $u$, codebook $c$ and server $n$ |
| $S_u^n$ | Offloading decision of device $u$ |
| $p_u$ | Transmission power of device $u$ |
| $\mathcal{U}_{\text{off}}^n$ | Set of devices offloading tasks to MEC server $n$ |
| $f_u^l$ | Local computing frequency of user $u$ |
| $f_u^n$ | Allocated edge computation resource for user $u$ at server $n$ |
| $f_n$ | Computation frequency of the MEC server $n$ |
| $t_u^l$ | Local computation time of user $u$ |
| $t_u^{\text{up}}$ | Uplink transmission time of user $u$ |
| $t_u^{\text{exe}}$ | Edge task execution time of user $u$ |
| $E_u^l$ | Local energy consumption of user $u$ |
| $E_u^e$ | Offloading energy consumption of user $u$ |
| $\beta_t$ | Devices' preference to time |
| $\beta_e$ | Devices' preference to energy |
| $L$ | The number of associated pilots for each SCMA codebook |
| $\kappa$ | Energy factor of the effective switching capacitance. |
| $G_u^l$ | Local utility consumption of user $u$ |
| $G_u^e$ | Edge utility consumption of user $u$ |
| $G_u$ | Utility consumption of user $u$ |

computation task $T_u$ is executed locally, the computation time and energy consumption can be expressed as

$$t_u^l = \frac{c_u}{f_u^l}, \tag{1}$$

$$E_u^l = \kappa (f_u^l)^2 c_u, \tag{2}$$

where $\kappa$ denotes the energy factor, determined by the effective switching capacitance. The local computing rate $f_u^l$ can be adjusted using DVFS technology [21] to optimize the execution time and energy consumption of IoT devices and the local computation policy is defined as $\mathcal{F}^l = \{f_u^l, u \in \mathcal{U}\}$.

### B. Task Offloading through SCMA

The devices adopt SCMA as the multiple access scheme to perform task offloading. At the SCMA transmitter, the binary bit stream is directly mapped to multidimensional codewords through SCMA coding. After physical resource mapping, codewords of multiple users are overlapped non-orthogonally in a sparse spreading way in the same time-frequency resources. The receiver adopts message passing algorithm (MPA) for joint multi-user detection and restores the original bit stream. The number of SCMA codewords can be much larger than the number of orthogonal resources, which enables SCMA to serve more users than orthogonal subcarriers. In the SCMA scheme, the modulation and spread spectrum steps are combined into a codeword mapping of the corresponding codebook, and the SCMA encoder can define as

$$f : \mathbb{B}^{\log_2(M)} \to \mathcal{X}, \mathbf{x} = f(\mathbf{b}), \tag{3}$$

where $\mathbf{b}$ is the input binary bit stream. $\mathbb{B}$ represents the set of binary numbers. $f$ indicates the mapping function between the binary bit stream and the SCMA codeword. $\mathbf{x}$ is a $K$-dimensional sparse codeword vector. $\mathcal{X}$ denotes the SCMA codebook of the user. The number of codewords in each codebook is $M$. $\mathcal{X} \in \mathbb{C}^K$ represents the complex set with dimension $K$. We denote the set of codebooks for every SCMA layer and subcarriers in every single-cell SCMA system as $\mathcal{C} = \{1, 2, \ldots, C\}$ and $\mathcal{K} = \{1, 2, \ldots, K\}$, respectively. The connection between SCMA codebooks and subcarriers can be expressed by the mapping matrix $\boldsymbol{F} = (\boldsymbol{F}_1, \boldsymbol{F}_2, \cdots, \boldsymbol{F}_C)$, where $\boldsymbol{F}_c = (a_{1,c}, a_{2,c}, \cdots, a_{K,c})$ is the indicator vector for the codebook $c$. Codewords in SCMA codebooks are mainly composed of non-zero elements and zero elements, which correspond to resource blocks. Let the number of non-zero

elements of the codeword be $R$. In fact, the user only transmits different modulation signals of the same bit sequence on the corresponding $R$ resource blocks. An example of the indicator matrix $\boldsymbol{F}$ with $C = 6, K = 4, R = 2$ is expressed as

$$\boldsymbol{F} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \tag{4}$$

If $a_{k,c}^n = 1$, it means that codebook $c$ occupies the $k$th subcarrier in BS $n$. $p_{u,c}^n$ denotes the transmit power of device $u$ to BS $n$ on codebook $c$ and it is allocated to subcarrier $k$ in a given proportion $\delta_{c,k}$, which satisfies $\sum_{\forall k \in c} \delta_{c,k}^n = 1$ [12]. Considering the characteristics of the multi-cell system, it should be noted that the number of codebooks is usually less than that of devices, which is different from the assumption in the single-cell SCMA network [19], [22]. The mapping between codebooks and subcarriers $a_{k,c}^n$ and the subcarrier power allocation proportion $\delta_{c,k}^n$ can be solved by the bidirectional matching principle and the water filling technique proposed in our previous work [23].

We denote $S_{u,c}^n$ as the offloading policy of users, which also incorporates multi-cell SCMA codebook allocation. If codebook $c$ is allocated to mobile device $u$ for offloading its computing task to the server $n$, then $S_{u,c}^n = 1$, otherwise, $S_{u,c}^n = 0$. Define $S_u^n = \sum_{c \in \mathcal{C}} S_{u,c}^n$ as the offloading decision of device $u$. And then the set of devices offloading tasks to MEC server $n$ can be shown as $\mathcal{U}_{\text{off}}^n = \{u \in \mathcal{U} \mid S_u^n = 1\}$. The offloading set of IoT devices is denoted as $\mathcal{U}_{\text{off}} = \bigcup_{n \in \mathcal{N}} \mathcal{U}_{\text{off}}^n$, and the computation offloading strategy can be defined as $\mathcal{S}$, a three dimensional matrix with each element $S_{u,c}^n$.

In the SCMA uplink model, we consider that multiple users in multi-cells may reuse the same codebook, and at most one codebook per user. There are $L$ associated pilots for each codebook in a contention transmission unit (CTU) in the adopted grant-free SCMA scheme [24]. As long as the pilot sequences are different, the SCMA receiver can detect the data stream carried by the same codebook [25]. The signal to interference plus noise ratio (SINR) of user $u$ on codebook $c$ in BS $n$ can be represented as [12]

$$\gamma_{u,c}^n = \frac{S_{u,c}^n \sum_{k \in \mathcal{K}} \delta_{c,k}^n p_{u,c}^n \left| h_{u,k}^n \right|^2}{I_{u,c}^n + \left(\sigma_{u,c}^n\right)^2}, \tag{5}$$

where $I_{u,c}^n = \sum_{n' \in \mathcal{N}/\{n\}} \sum_{u' \in \mathcal{U}/\{u\}} \sum_{k \in \mathcal{K}} \delta_{c,k}^{n'} p_{u',c}^{n'} \left| h_{u',k}^n \right|^2$ represents the inter-cell interference. $h_{u,k}^n$ and $(\sigma_{u,c}^n)^2$ are defined as the channel gain and noise power between the user $u$ and BS $n$ on the subcarrier $k$. From the SINR, the transmit rate of device $u$ in BS $n$ is expressed as

$$R_u^n = \sum_{c \in \mathcal{C}} \log_2 \left(1 + \gamma_{u,c}^n\right). \tag{6}$$

Let $p_u$ denote the transmission power of device $u$, then $p_{u,c}^n = p_u S_{u,c}^n$. We define the device power allocation strategy as $\mathcal{P} = \{p_u \mid u \in \mathcal{U}_{\text{off}}\}$. Therefore, the uplink transmission time of user $u$ when transferring the task description data $d_u$ to MEC server $n$ can be calculated as

$$t_u^{\text{up}} = \frac{d_u}{R_u^n}. \tag{7}$$

The offloading energy consumption of device $u$ can be expressed as

$$E_u^e = p_u t_u^{\text{up}}. \tag{8}$$

### C. MEC Execution Model

MEC servers adopt parallel processing for multiple computation tasks. The computation frequency of the MEC server $n$ is denoted as $f_n$ (in cycles/s), and the computation resource allocated to every associated user by the MEC server $n$ is quantified by $f_u^n$. The computation resource allocation strategy is defined as $\mathcal{F} = \{f_u^n, u \in \mathcal{U}, n \in \mathcal{N}\}$. Hence, the task execution time of user $u$ is calculated as

$$t_u^{\text{exe}} = \frac{c_u}{f_u^n}. \tag{9}$$

### D. Problem Formulation

We define the local utility consumption $G_u^l$ as the weighted sum of task delay and energy consumption,

$$G_u^l = \beta_t t_u^l + \beta_e E_u^l = \beta_t \frac{c_u}{f_u^l} + \beta_e \kappa (f_u^l)^2 c_u, \tag{10}$$

where $\beta_t \in [0, 1]$ and $\beta_e = 1 - \beta_t$ represent users' preference to time and energy. The total time when offloading computation tasks to MEC servers is calculated as

$$t_u^e = t_u^{\text{up}} + t_u^{\text{exe}} = \frac{d_u}{R_u^n} + \frac{c_u}{f_u^n}. \tag{11}$$

In practical applications such as target recognition, the amount of calculation results that need to be returned is three orders of magnitude smaller than the original task data and the downlink transmission latency is negligible. Then, the edge utility consumption can be obtained as

$$G_u^e = \beta_t t_u^e + \beta_e E_u^e = \beta_t \left( \frac{d_u}{R_u^n} + \frac{c_u}{f_u^n} \right) + \beta_e \frac{p_u d_u}{R_u^n}. \tag{12}$$

Hence, the utility consumption of device $u$ is expressed as

$$G_u = \sum_{n \in \mathcal{N}} (1 - \sum_{c \in \mathcal{C}} S_{u,c}^n) G_u^l + \sum_{n \in \mathcal{N}} \sum_{c \in \mathcal{C}} S_{u,c}^n G_u^e. \tag{13}$$

Under the constraints of the maximum latency of computation tasks, this paper considers local CPU rate adjustment, MEC server computing resource distribution, power allocation, multi-cell SCMA codebook allocation, and task offloading strategy to minimize the latency and energy consumption for IoT users. The above optimization problem can be expressed

as:

$$\min_{\mathcal{S},\mathcal{F}^l,\mathcal{P},\mathcal{F}} \sum_{u\in\mathcal{U}}\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} S^n_{u,c}\left[\beta_t\left(\frac{d_u}{R^n_u}+\frac{c_u}{f^n_u}\right)+\beta_e\frac{p_u d_u}{R^n_u}\right] + \sum_{u\in\mathcal{U}}\sum_{n\in\mathcal{N}}\left(1-\sum_{c\in\mathcal{C}} S^n_{u,c}\right)\left[\beta_t\frac{c_u}{f^l_u}+\beta_e\kappa\left(f^l_u\right)^2 c_u\right] \tag{14a}$$

$$s.t. \quad S^n_{u,c}\in\{0,1\}, \forall u\in\mathcal{U}, c\in\mathcal{C}, n\in\mathcal{N} \tag{14b}$$

$$\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} S^n_{u,c}\leqslant 1, \forall u\in\mathcal{U} \tag{14c}$$

$$\sum_{u\in\mathcal{U}} S^n_{u,c}\leqslant L, \forall c\in\mathcal{C}, \forall n\in\mathcal{N} \tag{14d}$$

$$\sum_{c\in\mathcal{C}} S^n_{u,c} t^e_u + \left(1-\sum_{c\in\mathcal{C}} S^n_{u,c}\right) t^l_u \leqslant t^{\max}_u, \forall u\in\mathcal{U} \tag{14e}$$

$$0\leqslant \sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} S^n_{u,c} p^n_{u,c}\leqslant p^{\max}_u, \forall u\in\mathcal{U} \tag{14f}$$

$$f^n_u>0, \forall u\in\mathcal{U}_{\text{off}}, n\in\mathcal{N} \tag{14g}$$

$$\sum_{u\in\mathcal{U}^n_{\text{off}}} f^n_u\leqslant f_n, \forall n\in\mathcal{N} \tag{14h}$$

$$f^l_{u,\min}\leqslant f^l_u\leqslant f^l_{u,\max}, \forall u\in\mathcal{U}. \tag{14i}$$

The constraints in the above problem can be explained as follows. Constraint (14b) is the binary variable that represents the task offloading decision and multi-cell SCMA codebook allocation. Constraint (14c) indicates that each user can offload its task to one edge server using one SCMA codebook. Constraint (14d) implies that the same codebook can be reused at most $L$ times by multiple devices. Constraint (14e) shows that the calculation time of local and edge execution cannot exceed the task tolerance latency $t^{\max}_u$. Constraint (14f) limits the transmision power for each device $u$. Constraints (14g) and (14h) ensure the computation frequency allocated to the associated devices is positive and does not exceed the server's computation capacity $f_n$. Constraint (14i) restricts the devices' computation frequency.

## III. Joint Task Offloading and Resource Allocation Scheme

The offloading decision $\mathcal{S}$ is coupled among the objective function and multiple constraints, which makes the problem (14) belong to the mixed-integer nonlinear programming (MINLP) and difficult to be solved. Hence, we can fix $S^n_{u,c}$ simplifies the problem (14) into two tractable subproblems, i.e., resource allocation and task offloading policy.

### A. Resource Allocation

*1) Local Frequency Optimization:* Firstly, the local CPU computing frequency of IoT devices can be adjusted to minimize the local utility consumption $G^l_u$. Considering the constraints $(14e)$ and $(14i)$, the optimization problem can be transformed into:

$$\min_{\mathcal{F}^l} \quad \sum_{u\in\mathcal{U}}\beta_t\frac{c_u}{f^l_u}+\beta_e\kappa\left(f^l_u\right)^2 c_u \tag{15a}$$

$$s.t. \quad \frac{c_u}{f^l_u}\leqslant t^{\max}_u, \forall u\in\mathcal{U} \tag{15b}$$

$$(14i).$$

Problem (15) is a standard convex optimization problem that can be settled utilizing CVX tools or the method we proposed in [19], which is omitted for brevity.

*2) Joint Power and Edge Computing Resource Allocation:* In this subsection, we will investigate the transmit power allocation for devices and computing resource allocation of MEC servers. Considering $S^n_{u,c}=1$ and the constraints related to $p_u$ and $f^n_u$, the optimization problem is expressed as follows,

$$\min_{\mathcal{P},\mathcal{F}} \quad \sum_{u\in\mathcal{U}_{\text{off}}}\beta_t\left(\frac{d_u}{R^n_u}+\frac{c_u}{f^n_u}\right)+\beta_e\frac{p_u d_u}{R^n_u} \tag{16a}$$

$$s.t. \quad \frac{d_u}{R^n_u}+\frac{c_u}{f^n_u}\leqslant t^{\max}_u, \forall u\in\mathcal{U}_{\text{off}} \tag{16b}$$

$$0<p_u\leqslant p^{\max}_u, \forall u\in\mathcal{U}_{\text{off}} \tag{16c}$$

$$(14g)\ (14h).$$

When $S^n_{u,c}=1$, the transmit rate of device $u$ can be simplified as

$$R^n_u=\log_2\left(1+\frac{p_u\sum_{k\in\mathcal{K}}\delta^n_{c,k}\left|h^n_{u,k}\right|^2}{I^n_{u,c}+\left(\sigma^n_{u,c}\right)^2}\right). \tag{17}$$

An achievable upper bound on $I^n_{u,c}$ is defined as

$$\tilde{I}^n_{u,c}\triangleq\sum_{n'\in\mathcal{N}/\{n\}}\sum_{u'\in\mathcal{U}/\{u\}}\sum_{k\in\mathcal{K}} p^{\max}_{u'}\delta^{n'}_{c,k}\left|h^n_{u',k}\right|^2. \tag{18}$$

Similar to [8], [26], we consider $\tilde{I}^n_{u,c}$ to be a good approximation of $I^n_{u,c}$. Firstly, the offloading policy $\mathcal{S}$ can select the appropriate relationship between users, SCMA codebooks, and BSs, to reduce the inter-cell interference $I^n_{u,c}$. Therefore, the interference with a small deviation has little impact on the uplink transmit rate and the system cost. Secondly, $p_{u'}$ is replaced by $p^{\max}_{u'}$ for the approximation of interference bound. They are usually in the same order of magnitude. According to 3GPP specification [27], the maximum power of IoT devices is below 23dBm, and thus the power deviation does not cause a large bias. Detailed simulation results to justify this simplification are shown in Fig. 11 in Section IV.

Let $\zeta^n_{u,c}\triangleq\frac{\sum_{k\in\mathcal{K}}\delta^n_{c,k}\left|h^n_{u,k}\right|^2}{\tilde{I}^n_{u,c}+\left(\sigma^n_{u,c}\right)^2}$, and then $R^n_u=\log_2\left(1+\zeta^n_{u,c}p_u\right)$. We can use the substitution method and introduce a new variable $\xi^n_u\triangleq 1/R^n_u$, then $p_u=\left(2^{1/\xi^n_u}-1\right)/\zeta^n_{u,c}$. The optimization problem (16)

can be rewritten as

$$\min_{\xi,\mathcal{F}} \quad \sum_{u\in\mathcal{U}_{\text{off}}} \beta_t \left( d_u \xi_u^n + \frac{c_u}{f_u^n} \right) + \beta_e d_u \xi_u^n \frac{2^{1/\xi_u^n}-1}{\zeta_{u,c}^n} \tag{19a}$$

$$s.t. \quad d_u \xi_u^n + \frac{c_u}{f_u^n} \leqslant t_u^{\max}, \forall u \in \mathcal{U}_{\text{off}} \tag{19b}$$

$$\xi_u^n \geqslant 1/\log_2\left(1+\zeta_{u,c}^n p_u^{\max}\right), \forall u \in \mathcal{U}_{\text{off}} \tag{19c}$$

$$(14g)\ (14h).$$

***Lemma* 1.** *Problem (19) is a convex optimization problem.*

*Proof:* See Appendix A. ∎

Since the constraint (19b) is coupled between $\xi_u^n$ and $f_u^n$, we can use the BCD algorithm to iteratively solve the optimization problem for each variable block of $\xi_u^n$ and $f_u^n$ while fixing the remaining block to the last updated value. Based on the BCD method, the problem (19) can be solved by addressing two subproblems iteratively, i.e., power scheduling with given $f_u^n$ and edge computing resource allocation with fixed $\xi_u^n$.

The power scheduling problem given $f_u^n$ can be written as

$$\min_{\xi} \quad \sum_{u\in\mathcal{U}_{\text{off}}} \beta_t d_u \xi_u^n + \beta_e d_u \xi_u^n \frac{2^{1/\xi_u^n}-1}{\zeta_{u,c}^n} \tag{20a}$$

$$s.t. \quad \xi_u^n \leqslant \frac{t_u^{\max} - c_u/f_u^n}{d_u}, \forall u \in \mathcal{U}_{\text{off}} \tag{20b}$$

$$(19c).$$

The root of equation $\beta_t d_u + \frac{\beta_e d_u}{\zeta_{u,c}^n}[2^{1/\xi_u^n}(1-\ln 2/\xi_u^n)-1]=0$ is denoted as $\xi_u^{n,0}$, which can be solved by the bisection method. From the Appendix A, we know that the objective function is convex and it decreases monotonically with the increase of $\xi_u^n$ when $\xi_u^n < \xi_u^{n,0}$, otherwise it increases monotonically. We define $\xi_u^{n,l} = 1/\log_2\left(1+\zeta_{u,c}^n p_u^{\max}\right)$ and $\xi_u^{n,h} = \left(t_u^{\max} - c_u/f_u^n\right)/d_u$. Hence, the optimal solution $\xi_u^{n^*}$ for problem (20) is

$$\xi_u^{n^*} = \begin{cases} \xi_u^{n,l}, & \xi_u^{n,0} \leqslant \xi_u^{n,l} \\ \xi_u^{n,0}, & \xi_u^{n,l} \leqslant \xi_u^{n,0} \leqslant \xi_u^{n,h} \\ \xi_u^{n,h}, & \xi_u^{n,0} \geqslant \xi_u^{n,h} \end{cases}. \tag{21}$$

The near-optimal power scheduling can be obtained by $p_u^* = \left(2^{1/\xi_u^{n^*}}-1\right)/\zeta_{u,c}^n$.

The edge computing resource allocation with fixed $\xi_u^n$ is rephrased as

$$\min_{\mathcal{F}} \quad \sum_{u\in\mathcal{U}_{\text{off}}} \beta_t c_u / f_u^n \tag{22a}$$

$$s.t. \quad f_u^n \geqslant \frac{c_u}{t_u^{\max} - d_u \xi_u^n} \tag{22b}$$

$$(14g)\ (14h).$$

It is obvious that problem (22) is convex and Slater's condition holds since the feasible region in problem (22) has a nonempty interior. Therefore, the Karush-Kuhn-Tucker (KKT) conditions are sufficient and necessary for optimality.

***Lemma* 2.** *The optimal edge computing resource allocation solution $f_u^{n^*}$ can be written as*

$$f_u^{n^*} = \begin{cases} \dfrac{c_u}{t_u^{\max} - d_u \xi_u^n}, & u \in \mathcal{U}_{off}^n \backslash \mathcal{U}_0 \\ \dfrac{(f_n - \sum_{u\in\mathcal{U}_{off}^n\backslash\mathcal{U}_0} f_u^{n^*})\sqrt{c_u}}{\sum_{u\in\mathcal{U}_0}\sqrt{c_u}}, & u \in \mathcal{U}_0, \end{cases} \tag{23}$$

*where $\mathcal{U}_0$ is defined as the set of $\mu_u = 0$.*

*Proof:* Derivations can be found in Appendix B. ∎

The optimal allocation strategy of edge computing resources can be obtained from (23). In each MEC server, the computation resources are evenly distributed with the square root of the workload $\sqrt{c_u}$. For those devices which can not satisfy the delay demand, $f_u^{n^*}$ is set to $c_u/(t_u^{\max} - d_u\xi_u^n)$. The rest of computational frequency is equally allocated w.r.t $\sqrt{c_u}$. The above process is repeated until all the devices meet the latency requirement, and then the optimal solution $f_u^{n^*}$ is obtained. In the joint power and computing resource allocation algorithm based on BCD, the transmit power and edge computing resource allocation vectors are updated iteratively, as summarized in Algorithm 1. By performing Algorithm 1 at each base station, the near-optimal transmission power $\mathcal{P}^*$ and edge resource allocation strategy $\mathcal{F}^*$ could be obtained.

**Algorithm 1:** Joint Power and Computing Resource Allocation Algorithm Based on Block Coordinate Descent

**Input**: $\beta_t$, $\beta_e$, $p_u^{\max}$, $f_n$, $\zeta_{u,c}^n$, $T_u$, $\mathcal{U}_{\text{off}}^n$
**Output**: Optimal transmit power $\mathcal{P}_n^*$ and edge computing resource allocation strategy $\mathcal{F}_n^*$ at BS $n$.

**Initialize:** Set $k=0$, convergence tolerance $\epsilon_1, \epsilon_2 > 0$, use the bisection method to get $\xi_u^{n,0}$ and find initial feasible solution $(\xi_u^n)^0 = \xi_u^{n,0}$.
**repeat**
  Compute $(f_u^n)^k$ by (23) with given $(\xi_u^n)^k$;
  Compute $(\xi_u^n)^{k+1}$ by (21) with given $(f_u^n)^k$;
  $k = k+1$;
**until** $\left\|(f_u^n)^k - (f_u^n)^{k-1}\right\| \leq \epsilon_1$, $\left\|(\xi_u^n)^k - (\xi_u^n)^{k-1}\right\| \leq \epsilon_2$;
Compute near-optimal transmit power $p_u^k = \left(2^{1/(\xi_u^n)^k}-1\right)/\zeta_{u,c}^n$.
Then, set $p_u^k$ and $(f_u^n)^k$, $\forall u \in \mathcal{U}_{\text{off}}^n$ as the near-optimal transmit power $\mathcal{P}_n^*$ and edge computing resource allocation strategy $\mathcal{F}_n^*$ at BS $n$.

### *B. Task Offloading and Multi-cell SCMA Codebook Allocation*

With a fixed task offloading and multi-cell SCMA codebook allocation decision $S_{u,c}^n$, we get the near-optimal solutions for the power and computation resources allocation $\mathcal{P}^*$, $\mathcal{F}^*$, $\mathcal{F}^{l^*}$. Therefore, the optimization problem is formulated as

$$\min_{\mathcal{S}} \sum_{u\in\mathcal{U}}\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} S_{u,c}^n G_u^e\left(\mathcal{P}^*, \mathcal{F}^*\right) + \sum_{u\in\mathcal{U}}\sum_{n\in\mathcal{N}} \left(1 - \sum_{c\in\mathcal{C}} S_{u,c}^n\right) G_u^l\left(\mathcal{F}^{l^*}\right) \tag{24}$$

$$s.t. \quad (14b)\ (14c)\ (14d)\ (14e).$$

Using the exhaustive search method on all possible task offloading strategies and multi-cell SCMA codebook allocation is a direct and brute approach for problem (24). However, the total number of candidate solutions is $2^{U\times N\times C}$, and the exponential complexity makes the exhaustive search method impractical. Therefore, a low complexity algorithm based on improved simulated annealing is proposed.

The simulated annealing algorithm is derived from the process of crystal cooling. When the solid is heated and then cooled, as the temperature decreases slowly, the atoms are arranged into a certain shape, forming regular crystals with high density and low energy, which corresponds to the global near-optimal solution in the algorithm. The SA algorithm consists of two parts: the metropolis criterion [28] and the annealing process. Instead of using fully determined rules, the metropolis criterion accepts new states with probability. Specifically, the improved simulated annealing algorithm would update new solutions from the solutions of the last iteration in the following four ways:

(1) Change task offloading decision for device $u$.

(2) Randomly select another MEC server of the three closest MEC servers and find an SCMA codebook that has been allocated less than $L$ devices.

(3) Select another SCMA codebook that has been allocated less than $L$ devices.

(4) Exchange the MEC server and SCMA codebook policy with another device.

If the value of the new objective function ($G^{\text{new}}$) is less than the previous value ($G^{\text{old}}$), then the new solution ($\mathcal{S}^{\text{new}}$) is accepted. Otherwise, the algorithm accepts $\mathcal{S}^{\text{new}}$ as the new solution with probability $e^{-\Delta G/T}$. Finally, as the temperature decreases, the algorithm gradually converges to the an approximate optimal solution. At each iteration, it needs to perform Algorithm 1 with $\mathcal{S}^{\text{new}}$ in each base station, and obtain the near-optimal transmission power $\mathcal{P}^*$ and edge resource allocation strategy $\mathcal{F}^*$. The specific steps for task offloading and multi-cell SCMA codebook allocation based on SA (TOCA-SA) are described in Algorithm 2.

It should be noted that Algorithm 2 is applicable in the centralized situation, i.e., with a central controller knowing all the states in the environment. However, it is difficult to obtain global information in a timely manner in the highly dynamic environment of IoT devices. In practical application, due to the variability of channel states and task generation, it is often necessary to use the proposed algorithm to decide task offloading and multi-cell SCMA codebook allocation for every single state. In this case, Algorithm 2 is optimized for each time slot, and thus brings high complexity. Therefore, aiming at the above two shortcomings, we put forward a distributed deep reinforcement learning (DRL) algorithm to solve computation offloading policy and SCMA codebook allocation. Through centralized training and distributed execution, it can effectively solve the problem of partial state observation and timely decision-making. The proposed online algorithm avoids solving the optimization problem for each state, thus greatly reducing the complexity of determining the offloading solutions and SCMA codebook selections for different channel implementations and task generations.

**Algorithm 2:** Task Offloading and Multi-cell SCMA Codebook Allocation Based on Simulated Annealing

**Input**: $\mathcal{U}$, $\mathcal{C}$, $\mathcal{N}$, $\mathcal{K}$, $\beta_t$, $p_u^{\max}$, $f_n$, $T_u$, $G_u^{l^*}$, $T$, Minimum temperature $T_{\min}$, Reduction factor $\alpha$

**Output**: The task offloading and multi-cell SCMA codebook allocation decision $\mathcal{S}^*$

**Initialize:** Randomly generate an initial feasible solution $\mathcal{S}^{\text{old}}$ and calculate $G^{\text{old}} = \sum_{u\in\mathcal{U}} G_u$.
**while** $T > T_{\min}$ **do**
  Generate new solutions $\mathcal{S}^{\text{new}}$ from $\mathcal{S}^{\text{old}}$ in one of several following ways based on the probability $rand \in (0, 1)$.
  **if** $rand < 0.2$ **then**
    Change task offloading decision for device $u$
  **else**
    **if** $rand < 0.4$ **then**
      Randomly select another MEC server of the three closest servers and find an SCMA codebook that has been allocated less than $L$ devices.
    **else**
      **if** $rand < 0.8$ **then**
        Select another SCMA codebook that has been allocated less than $L$ devices.
      **else**
        Exchange the MEC server and SCMA codebook policy with another device.
      **end**
    **end**
  **end**
  Perform Algorithm 1 with $\mathcal{S}^{\text{new}}$ in each base station, and obtain the near-optimal transmission power $\mathcal{P}^*$ and edge resource allocation strategy $\mathcal{F}^*$.
  Calculate $G^{\text{new}}$, and $\Delta G = G^{\text{new}} - G^{\text{old}}$.
  **if** $\Delta G \leqslant 0$ **then**
    $\mathcal{S}^{\text{old}} = \mathcal{S}^{\text{new}}$, $G^{\text{old}} = G^{\text{new}}$
  **else**
    **if** $e^{-\Delta G/T} > rand(0, 1)$ **then**
      $\mathcal{S}^{\text{old}} = \mathcal{S}^{\text{new}}$, $G^{\text{old}} = G^{\text{new}}$
    **end**
  **end**
  $T = \alpha * T$
**end**

In the proposed reinforcement learning framework, each IoT device is defined as an agent that learns the computation offloading strategy and SCMA codebook allocation through interacting with the environment. On the basis of the original optimization problem (24), the state space, action space and reward function are designed as follows.

*State space*: Denote $\boldsymbol{o}_{u,t}$ as the environment state observed by IoT device $u$ at time slot $t$. The observed state of each device includes the distances from each BS $\boldsymbol{d}_u(t) = \{d_{u,n}(t), n \in \mathcal{N}\}$, channel gains of subcarriers in SCMA $\boldsymbol{h}_u(t) = \{h_{u,k}(t), k \in \mathcal{K}\}$, and the current task $T_u(t)$, Hence, the partial observed state of agent $u$ is expressed as

$\boldsymbol{o}_{u,t} = \{\boldsymbol{d}_u(t), \boldsymbol{h}_u(t), T_u(t)\}$ with size $N + K + 3$.

*Action space*:The action $\boldsymbol{a}_{u,t}$ of IoT device $u$ includes the decision of whether to carry out computation offloading ($s_u(t) \in \{0, 1\}$), where to offload ($b_u(t) \in \{1, ..., N\}$) and the selection of an SCMA codebook ($c_u(t) \in \{1, ..., C\}$). The action is given as $\boldsymbol{a}_{u,t} = \{s_u(t), b_u(t), c_u(t)\}$ with size 3. Note that $S^n_{u,c}$ in problem (24) can be easily get by $\boldsymbol{a}_{u,t}$.

*Reward*: $r_{u,t}$ is the reward after taking the action $\boldsymbol{a}_{u,t}$ at time slot $t$ and it is defined as the negative sum of task latency and energy consumption, i.e., the negative of optimization objective in problem (24), expressed as

$$r_{u,t} = \begin{cases} -\left[\beta_t(t^l_u + t^e_u) + \beta_e(E^l_u + E^e_u)\right], & t^e_u + t^l_u \leqslant t^{\max}_u \\ -p, & otherwise. \end{cases} \tag{25}$$

When the IoT device can not meet the delay requirement of its task, i.e., constraint (14e) is not satisfied, the agent gets punishment $p$ (a relatively large value).

The proposed online reinforcement learning framework is based on MADDPG with centralized training and decentralized execution. On the basis of actor-critic framework, each IoT device $u$ consists of an actor network and a critic network. Both the actor network and the critic network contain two networks with the same structure, i.e., evaluation network and target network. The parameters of target network remain fixed for several steps and then update according to the weights of the evaluation network. The fixed target network reduces the correlation between the target and the estimated value. Actor network can make decisions based on partial environment states observed by IoT devices. Critic network is centralized and it can obtain global information (including global environment states and all actions of agents), which gives the corresponding Q value $Q(\boldsymbol{x}, \boldsymbol{a}_1, ..., \boldsymbol{a}_U), \boldsymbol{x} = [\boldsymbol{o}_1, ..., \boldsymbol{o}_U]$. It should be noted that the centralized training stage is offline and all the actions of agents can be acquired centrally at BS for training. Therefore, the centralized training process can solve the instability problem of multiagent environment to a certain extent. In the decentralized execution phase, it only propagates forward based on partial observations. Therefore, the trained actor network can be deployed on IoT devices.

*Centralized training*: The network parameters of IoT devices are defined as $\theta = [\theta_1, \cdots, \theta_U]$ and deterministic policies can be expressed as $\mu = [\mu_{\theta_1}, \cdots, \mu_{\theta_U}]$. For ease of writing, $\mu_u$ is short for $\mu_{\theta_u}$. In the centralized training process of critic network, the update method is based on temporal difference (TD) error and then the loss function can be written as

$$L(\theta_u) = \mathbb{E}_{\boldsymbol{x},a,r,\boldsymbol{x}'}\left[\left(Q^{\mu}_u(\boldsymbol{x}, a_1, \cdots, a_U) - y\right)^2\right], \tag{26}$$

where $y = r_u + \gamma Q^{\mu'}_u(x', a'_1, \cdots, a'_U)\big|_{a'_u = \mu'_u(o_u)}$ and $\gamma$ is the discount factor. $\mu'$ is the parameter of the corresponding target network. The actor network update the parameters in the direction of $\nabla_{\theta_u} J$ and the gradient can be expressed as,

$$\begin{aligned} &\nabla_{\theta_u} J(\mu_u) \\ &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{a}\sim\mathcal{D}}\left[\nabla_{\theta_u}\mu_u(o_u)\nabla_{a_u} Q^{\mu}_u(\boldsymbol{x}, a_1, \cdots, a_U)\big|_{a_u=\mu_u(o_u)}\right], \end{aligned} \tag{27}$$

where $\mathcal{D}$ is the experience replay buffer, containing the series of observation data $(\boldsymbol{x}, a_1, \cdots, a_U, r_1, \cdots, r_U, \boldsymbol{x}')$. Experience replay method can be used to randomize data and eliminate correlations in the observations.

*Decentralized execution*: In the decentralized execution phase, the trained actor network is assigned to the IoT device side to make online decisions. Action is performed based on the observed state of the device,

$$a_u = \mu(o_u|\theta_u). \tag{28}$$

Since each agent only relies on local observations to make decisions, it does not need a complex communication network to keep in touch with other agents. Therefore, it can be well adapted to practical applications. The specific steps of the training algorithm for joint task offloading and SCMA codebook allocation (TOCA-MADDPG) are presented in Algorithm 3, and the execution algorithm on the IoT device is shown in Algorithm 4.

Different from MADDPG methods in other papers [29], [30], the TOCA-MADDPG algorithm proposed in this paper solves the problem that the reward cannot converge when there are too many agents and the environment is unstable. Due to the characteristics of the multi-cell SCMA-MEC networks environment, the distances between IoT devices and base stations play a dominant role in the channel gain, and when the channel gain is larger, the IoT device has a stronger willingness to perform computation offloading. As the cell number further increases ($N > 3$), we limit the agent's second action $b_u(t)$, i.e., which base station or MEC server to choose as the offloading destination, to the three nearest base stations. By doing so, the state size of every agent is 10 and only $1 + 3 * 6 = 19$ actions can be selected. In this case, the uncertainty of the environment and the convergence complexity of the algorithm can be greatly reduced. Therefore, the proposed TOCA-MADDPG algorithm can still converge well in the case of multiple agents.

### C. Analysis of Special Cases

The minimization problem of the weighted energy consumption and latency is studied in problem (14). In this subsection, we analyze two special cases of the above problem. The analysis explains the generality and applicability of the studied problem and proposed algorithms. When the weighting factor $\beta_t$ is equal to 1, the latency minimization problem can be formulated as follows,

$$\begin{aligned} \min_{\mathcal{S},\mathcal{F}^l,\mathcal{P},\mathcal{F}} & \sum_{u\in\mathcal{U}}\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} S^n_{u,c}\left(\frac{d_u}{R^n_u} + \frac{c_u}{f^n_u}\right) \\ & + \sum_{u\in\mathcal{U}}\sum_{n\in\mathcal{N}}\left(1 - \sum_{c\in\mathcal{C}} S^n_{u,c}\right)\frac{c_u}{f^l_u} \\ s.t. \quad & (14b)\ (14c)\ (14d)\ (14e)\ (14f)\ (14g)\ (14h)\ (14i). \end{aligned} \tag{29}$$

**Algorithm 3:** Centralized Training in TOCA-MADDPG Algorithm

**Input**: The environment parameters: $\mathcal{U}$, $\mathcal{N}$, $\mathcal{C}$, $\mathcal{K}$, $\beta_t$, $p_u^{\max}$, $f_n$, $T_u$, $G_u^{l^*}$; MADDPG training parameters: $E, T_S, \alpha_{\text{actor}}, \alpha_{\text{critic}}, \mathcal{D}, \gamma, \tau$, minibatch size $I$, noise decay rate $\alpha_n$.

**Output**: The trained weights $\theta$ of actor networks for every agent.

**Initialize:** Randomly set the weights of actor network and critic network for every agent. Empty the replay buffer $\mathcal{D}$. Initialize a Gaussian process with $mean = 0$ and $var = 2$.
**for** *episode*=$1, 2, ..., E$ **do**
  Reset the simulation parameters of SCMA-enabled multi-cell edge computing networks.
  IoT devices observes an initial state $\boldsymbol{x} = [\boldsymbol{o}_1, ..., \boldsymbol{o}_U]$.
  **for** *time slot*=$1, 2, ..., T_S$ **do**
    for each IoT device $u$, select action $a_u = \mu_u(\boldsymbol{o}_u) + n_u$, add random Gaussian noise $n_u \sim \mathbb{N}(0, var)$ to ensure exploration.
    $var = var * \alpha_n$.
    Execute actions $\boldsymbol{a} = [a_1, \cdots, a_U]$ for every agent, observe the reward $\boldsymbol{r} = [r_1, \cdots, r_U]$ by (25) and next state $\boldsymbol{x}\_$.
    Store experience $(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{x}')$ in $\mathcal{D}$.
    $\boldsymbol{x} = \boldsymbol{x}'$
    **for** *agent*=$1, 2, ..., U$ **do**
      Randomly sample $I$ transitions as a train minibatch from $\mathcal{D}$.
      Update critic network by minimizing the loss $L(\theta_u)$ in (26).
      Update actor network using the policy gradient in the direction of $\nabla_{\theta_u} J(\mu_u)$ in (27).
      Update target networks by $\theta'_u = \tau\theta_u + (1-\tau)\theta'_u$.
    **end**
  **end**
**end**

**Algorithm 4:** Decentralized Execution in TOCA-MADDPG Algorithm

**Input**: Time slots $T_S$. Current states $\boldsymbol{d}_u, \boldsymbol{h}_u, T_u$. The trained weights $\theta$ of actor networks for every agent.

**Output**: The actions $\boldsymbol{a}$ of all IoT devices

**for** *time slot*=$1, 2, ..., T_S$ **do**
  **for** *agent*=$1, 2, ..., U$ **do**
    Select action $a_u = \mu_u(\boldsymbol{o}_u)$ based on the observation $\boldsymbol{o}_u$.
  **end**
  Environment states change to $\boldsymbol{x}'$ based on the actions $\boldsymbol{a}$ selected by agents.
  Every agent gets reward $\boldsymbol{r}$ by (25) and observates next state.
**end**

Similarly, when the parameter $\beta_e = 1$, the energy minimization problem can be rewritten as,

$$\begin{aligned}\min_{\mathcal{S},\mathcal{F}^l,\mathcal{P}} & \sum_{u\in\mathcal{U}}\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} S_{u,c}^n \frac{p_u d_u}{R_u^n} \\ & + \sum_{u\in\mathcal{U}}\sum_{n\in\mathcal{N}}\left(1-\sum_{c\in\mathcal{C}} S_{u,c}^n\right)\kappa\left(f_u^l\right)^2 c_u \\ s.t. \quad & (14b)\ (14c)\ (14d)\ (14e)\ (14f)\ (14i).\end{aligned} \tag{30}$$

These two problems are special cases of the problem studied in this paper. Obviously, the algorithms proposed can be directly used to solve the problems of minimizing task execution delay and energy consumption.

### *D. Extension to Partial Offloading*

This subsection extends the proposed algorithms to partial offloading. In this case, we assume that task $T_u$ can be divided into two arbitrarily sized blocks, one of which is executed locally by the UE itself and the other ($\delta_u$) can be offloaded to the edge server.

When the number of CPU cycles processed locally is $(1-\delta_u)c_u$, the execution time and energy consumption can be expressed as $t_u^l = (1-\delta_u)c_u/f_u^l$ and $E_u^l = \kappa(f_u^l)^2(1-\delta_u)c_u$, respectively. When computation offloading is performed, the total time spent can be given as $t_u^c = \delta_u d_u/R_u^n + \delta_u c_u/f_u^n$, and the energy consumption is expressed as $E_u^c = \delta_u p_u d_u/R_u^n$. Since parallel computing is performed simultaneously on the device and the MEC server, the delay of the overall task should be the maximum value of the two, i.e., $\max\left\{(1-\delta_u)c_u/f_u^l, \delta_u d_u/R_u^n + \delta_u c_u/f_u^n\right\}$. In addition, the overall energy consumption can be expressed as $\kappa(f_u^l)^2(1-\delta_u)c_u + \delta_u p_u d_u/R_u^n$.

Therefore, similar to binary offloading, the tradeoff problem between latency and energy consumption can be formulated as

$$\begin{aligned}\min_{\mathcal{S},\delta,\mathcal{F}^l,\mathcal{P},\mathcal{F}} & \sum_{u\in\mathcal{U}}\beta_t \max\left\{\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} S_{u,c}^n\delta_u\left(\frac{d_u}{R_u^n}+\frac{c_u}{f_u^n}\right), (1-\delta_u)\frac{c_u}{f_u^l}\right\} \\ & + \sum_{u\in\mathcal{U}}\beta_e\left[\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} S_{u,c}^n\delta_u\frac{p_u d_u}{R_u^n} + (1-\delta_u)\kappa\left(f_u^l\right)^2 c_u\right] \qquad (31a) \\ s.t. \quad & \delta_u \in [0,1], \forall u \in \mathcal{U} \qquad (31b) \\ & \max\left\{\delta_u t_u^e, (1-\delta_u)\, t_u^l\right\} \leqslant t_u^{\max}, \forall u \in \mathcal{U} \qquad (31c) \\ & (14b)\ (14c)\ (14d)\ (14f)\ (14g)\ (14h)\ (14i).\end{aligned}$$

The constraint (31b) represents the offloading policy $\delta_u$ is a continuous variable within $[0, 1]$. Constraint (31c) implies that the task execution time cannot exceed the maximum task execution latency $t_u^{\max}$. Note that $S_{u,c}^n$ in this section only represents the relationship between IoT devices, BS, and multi-cell SCMA codebook allocation. $\delta_u$ stands for offloading policy.

When fixing $S_{u,c}^n$, the optimization problem is transformed

as

$$\min_{\delta,\mathcal{F}^l,\mathcal{P},\mathcal{F}} \sum_{u\in\mathcal{U}} \beta_t \max\left\{\delta_u\left(\frac{d_u}{R_u^n}+\frac{c_u}{f_u^n}\right),(1-\delta_u)\frac{c_u}{f_u^l}\right\} + \sum_{u\in\mathcal{U}} \beta_e\left[\delta_u\frac{p_u d_u}{R_u^n}+(1-\delta_u)\kappa\left(f_u^l\right)^2 c_u\right] \quad (32a)$$

$$s.t. \quad 0\leqslant p_{u,c}^n \leqslant p_u^{\max}, \forall u\in\mathcal{U} \quad (32b)$$

$$(14g)\ (14h)\ (14i)\ (31b)\ (31c).$$

Using the same idea in subsection III-A, let $\xi_u^n = 1/R_u^n$, the optimization problem (32) can be rewritten as

$$\min_{\delta,\mathcal{F}^l,\xi,\mathcal{F}} \sum_{u\in\mathcal{U}} \beta_t \max\left\{\delta_u\left(d_u\xi_u^n+\frac{c_u}{f_u^n}\right),(1-\delta_u)\frac{c_u}{f_u^l}\right\} + \sum_{u\in\mathcal{U}} \beta_e\left[\delta_u d_u\xi_u^n\frac{\left(2^{1/\xi_u^n}-1\right)}{\zeta_{u,c}^n}+(1-\delta_u)\kappa\left(f_u^l\right)^2 c_u\right] \quad (33a)$$

$$s.t. \quad \delta_u\left(d_u\xi_u^n+\frac{c_u}{f_u^n}\right)\leqslant t_u^{\max}, \forall u\in\mathcal{U} \quad (33b)$$

$$(1-\delta_u)c_u/f_u^l \leqslant t_u^{\max}, \forall u\in\mathcal{U} \quad (33c)$$

$$\xi_u^n \geqslant 1/\log_2\left(1+\zeta_{u,c}^n p_u^{\max}\right), \forall u\in\mathcal{U} \quad (33d)$$

$$(14g)\ (14h)\ (14i)\ (31b).$$

Even if the transformed problem (33) is still non-convex, it is noted that when $\delta$ is fixed, the problem is a standard convex optimization problem, because the objective function is the maximum of two convex functions [31]. The interior-point method and CVX tools can be used to solve it. When $\mathcal{F}^l, \xi, \mathcal{F}$ are fixed, that problem can be transformed to a linear programming problem w.r.t $\delta$, which is easy to be settled. Then, the near-optimal solutions can be solved by an alternating convex search algorithm, shown in Algorithm 5. When $\delta, \mathcal{F}^l, \mathcal{P}, \mathcal{F}$ are solved, the improved simulated annealing algorithm proposed in Algorithm 2 can be used to solve $\mathcal{S}$. For the dynamic environment, the computation offloading decision $s_u(t) \in \{0,1\}$ only needs to be changed into $s_u(t) \in [0,1]$ in the action space, and then Algorithm 3 can solve the partial offloading case.

**Algorithm 5:** Resource Allocation Algorithm and Partial Offloading Policy Based on Alternate Convex Search

**Input**: $\beta_t$, $\beta_e$, $T_u$, $p_u^{\max}$, $f_n$, $\zeta_{u,c}^n$, $f_{u,\min}^l$, $f_{u,\max}^l$, convergence tolerance $\epsilon$.

**Output**: Optimal resource allocation $\mathcal{F}^{l^*}, \mathcal{P}^*, \mathcal{F}^*$ and partial task offloading strategy $\delta^*$.

1 **Initialize:** Set iterxation index $k=1$ and the initial value of $\delta^1$ equals 0.5.
2 **repeat**
3 Substitute $\delta^k$ into the problem (33) to obtain $\left((f^l)^{k+1},(\xi_u^n)^{k+1},(f_u^n)^{k+1}\right)$ by solving the convex optimization problem using the interior point method.
4 Update $\delta^{k+1}$ based on $\left((f^l)^{k+1},(\xi_u^n)^{k+1},(f_u^n)^{k+1}\right)$ by solving the linear programming problem.
5 $k=k+1$;
6 $G(k-1)=G_u\left(\delta_u^{k-1},(f^l)^{k-1},(\xi_u^n)^{k-1},(f_u^n)^{k-1}\right)$;
7 $G(k)=G_u\left(\delta_u^k,(f^l)^k,(\xi_u^n)^k,(f_u^n)^k\right)$;
8 **until** $\|G(k)-G(k-1)\|\leq\epsilon$;
9 Compute near-optimal transmit power $p_u^*=\left(2^{1/(\xi_u^n)^k}-1\right)/\zeta_{u,c}^n$.

### *E. Computational Complexity Analysis*

The computational complexity of the proposed algorithms is discussed in this subsection. Firstly, the complexity of Algorithm 1 is dominated by lines 1, 3 and 4. Define the set of users offloading to BS $n$ as $U_{\text{off}}^n = |\mathcal{U}_{\text{off}}^n|$. The first line adopts the bisection search method for solving $\xi_u^{n,0}$, and the complexity is on the order of $\mathcal{O}\left(\log_2(U_{\text{off}}^n)\right)$. The complexity of Line 3 and Line 4 by calculating $\xi_u^n, f_u^n$ using (21) and (23) is $\mathcal{O}\left(2U_{\text{off}}^n\right)$ and $\mathcal{O}\left(U_{\text{off}}^n\right)$, respectively. The number of iterations of Algorithm 1 is expressed as $J_1$, and then the complexity of Algorithm 1 is $\mathcal{O}\left(\log_2(U_{\text{off}}^n)+J_1\left(3U_{\text{off}}^n\right)\right)$.

In view of the complexity of Algorithm 2, it is decided by new policy $\mathcal{S}^{\text{new}}$ generation and performing Algorithm 1 with $\mathcal{S}^{\text{new}}$. The complexity of generating $\mathcal{S}^{\text{new}}$ for lines $3-16$ is on the order of $\mathcal{O}\left(UCN\right)$. Line 17 runs Algorithm 1 in each base station, the complexity is $\mathcal{O}\left(\sum_{n=1}^N\left(\log_2(U_{\text{off}}^n)+J_1\left(3U_{\text{off}}^n\right)\right)\right)$. The number of iterations is calculated as $\log_\alpha(T_{\min}/T)$. Therefore, the complexity of Algorithm 2 is denoted as $\mathcal{O}\left(\log_\alpha(T_{\min}/T)UCN\sum_{n=1}^N\left(\log_2(U_{\text{off}}^n)+J_1\left(3U_{\text{off}}^n\right)\right)\right)$. If the exhaustive method for task offloading and SCMA codebook allocation is used, the complexity is $\mathcal{O}\left(2^{UCN}\sum_{n=1}^N\left(\log_2(U_{\text{off}}^n)+J_1\left(3U_{\text{off}}^n\right)\right)\right)$. Therefore, compared with the exhaustive method, the proposed algorithm greatly reduces the computational complexity and more detailed iterations results are shown in Section IV.

The complexity of Algorithm 3 lies in the training of actor network and critic network for each agent, from label $12-15$. For actor network, the complexity of one training is $\mathcal{O}\left(|\boldsymbol{o}_{u,t}|\eta+\eta|\boldsymbol{a}_{u,t}|\right)$ [32], where $\eta$ is the hidden layer size of the network, $|\boldsymbol{o}_{u,t}| = N+K+3$ denotes the observation size and $|\boldsymbol{a}_{u,t}| = 3$ shows the action size. The complexity of training for critic network is $\mathcal{O}\left(U(|\boldsymbol{o}_{u,t}|+|\boldsymbol{a}_{u,t}|)\eta+\eta\right)$. Therefore, the computational complexity of Algorithm 3 is calculated and simplified as $\mathcal{O}\left(ET_S U(|\boldsymbol{o}_{u,t}|+|\boldsymbol{a}_{u,t}|)\eta\right)$. The computational complexity of Algorithm 4 depends on step 3, and the complexity of obtaining action by a IoT device is $\mathcal{O}\left(|\boldsymbol{o}_{u,t}|\eta+\eta|\boldsymbol{a}_{u,t}|\right)$. Therefore, the overall complexity of Algorithm 4 is expressed as $\mathcal{O}\left(T_S U\left(|\boldsymbol{o}_{u,t}|\eta+\eta|\boldsymbol{a}_{u,t}|\right)\right)$.

The complexity of Algorithm 5 is dominated by interior point method and linear programming method using CVX tools in the third and fourth lines, respectively. Its computational complexity is related to the corresponding optimization constraints and decision variables. The original convex problem in line 3 has $4U+UN+N$ linear matrix inequality (LMI) constraints of size 1 and $2U+UN$ decision variables. Therefore, the computational complexity in line 3 is $\mathcal{O}\left(\sqrt{4U+UN+N}(2U+UN)^3\right)$. There are $3U$ LMI constraints of size 1 and $U$ decision variables of the original convex problem in line 4. The complexity in line 4

is $\mathcal{O}\left(\sqrt{3U}U^3\right)$. Define the number of iterations in algorithm 3 as $J_3$. Then the complexity of Algorithm 5 is expressed as $\mathcal{O}\left(J_3(\sqrt{4U+UN+N}(2U+UN)^3+\sqrt{3U}U^3)\right)$.

## IV. Simulation Results and Analysis

In this section, representative simulation results are shown to evaluate the performance of the multi-cell SCMA-enabled MEC design. Specifically, $N$ BSs and $U$ users are randomly distributed in an area, and the adjacent BSs are set 100m apart from each other. The number of SCMA codebooks is set as $C = 6$. For each codebook, the number of associated pilot sequences is $L = 2$ [25]. The channels between the users and BSs are generated by a distance dependent path loss, modeled as $PL(\text{dB}) = 140.7 + 36.7log_{10}(d)_{[km]}$ [8] with 8dB log-normal multipath shadowing. The maximum transmit power of users $p_u^{\max}$ is 23dBm. We set the system bandwidth $B$ as 20MHz and the number of subcarriers as 4. Every BS is equipped with a 20GHz MEC server. The task description data $d_u$ and the workload $c_u$ are randomly distributed in $[300, 1200]$KB and $[0.2, 1]$Gcycles, respectively [33]. The maximum latency $t_u^{\max}$ is between 0.5s and 2s. The terminal device can adjust its computation rate as 0.2GHz∼1GHz and $\kappa = 5 \times 10^{-27}$ [34]. The termination temperature and cooling speed of the TOCA-SA algorithm are set as $T_{\min} = 0.0001$ and $\alpha = 0.98$. In the TOCA-MADDPG algorithm, the actor and critic network are implemented by five fully connected layers, including one input layer, three hidden layers (256, 128 and 64 neurons), and one output layer. The training parameters of the proposed TOCA-MADDPG algorithm are set as follows: train episodes $E = 1000$, time slots $T_S = 100$, minibath size $I = 64$, experience replay buffer size $|\mathcal{D}| = 3000$, learning rate of actor network $\alpha_{\text{actor}} = 0.001$ and critic network $\alpha_{\text{critic}} = 0.002$, reward discount $\gamma = 0.9$, soft update $\tau = 0.01$, noise $var = 2$ and noise decay rate $\alpha_n = 0.99$. The simulation results were obtained on a laptop equipped with R9-5900HX, RTX3080 and 32GB RAM.

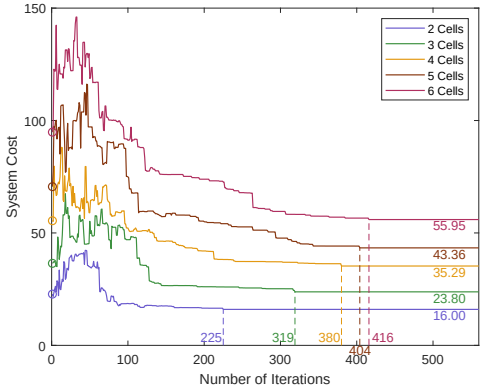
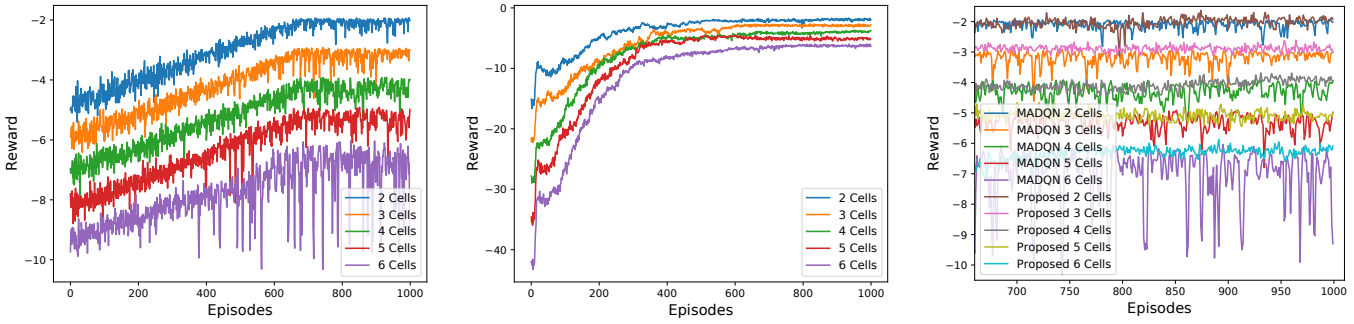



Fig. 2. Trend of the system cost versus the number of iterations and cells.

The convergence performance of the proposed task offloading and multi-cell SCMA codebook allocation based on an improved simulated annealing algorithm is illustrated in Fig. 2. We consider that each BS can serve an average of 6 devices, and when the cell number increases from 2 to 6, the number of devices grows from 12 to 36. In all cases, the utility value experiences a period of fluctuation due to label 13 in Algorithm 2, then decreases, and finally converges with the increase of iterations. It can be seen from the figure that the convergence can be achieved within 500 iterations in all cells. When the number of cells grows from 2 to 6, the number of iterations reaching equilibrium goes from 225 to 416. Compared with the exhaustive search method, which requires $2^{12\times2\times6} \sim 2^{36\times6\times6}$ iterations [8], the complexity of proposed algorithm is significantly reduced.

The training process of the TOCA-MADDPG algorithm is shown in Fig. 3, in which the proposed algorithm is compared with the multi-agent deep Q-learning networks (MADQN) method [35]. In Fig. 3(a), MADQN is applied to SCMA-MEC networks. Each IoT device is regarded as an agent, and the Q value is estimated through DNN networks. In MADQN, the reward is not only related to the action of a single agent but also depends on that of other agents. The joint action of multiple IoT devices affects the environment, which moves to the next state and receives corresponding rewards for each agent. When the number of cells is relatively small, the MADQN algorithm can gradually converge to a stable value through experience replay and interactive training with the environment. However, when the cell number gets larger, such as 6, the environment becomes unstable due to more devices taking action, resulting in the failure of reward convergence. Fig. 3(b) shows the obtained reward of the proposed TOCA-MADDPG algorithm during the training. As can be seen from the figure, the reward can converge to a relatively stable value in all the cases of cell numbers. Note that the reward is particularly low during the exploratory stage of TOCA-MADDPG. This is because a large number of IoT devices choose action 0, i.e, local computing, resulting in penalty $p$ in (14) for task failure. Through continuous training with the environment, the final reward is greatly increased. At this time, the trained computation offloading and codebook allocation strategy is effective, and the trained actor network can be directly deployed in IoT devices. Fig. 3(c) illustrates the difference in convergence values between MADQN and the proposed TOCA-MADDPG. It can be seen from the figure that the convergent rewards of the two algorithms are basically the same, while the reward of TOCA-MADDPG is relatively higher and more stable when the number of cells increases, indicating that the proposed algorithm shows more advantages.

Fig. 4 shows the effect of the proposed TOCA-MADDPG algorithm limiting the involving base stations on the reward convergence performance when $N > 3$. As can be seen from the figure, with the initial increase in the training episodes, the reward of the proposed algorithm and the traditional MADDPG algorithm grows through continuous training. The reason for the increase in rewards is that the agents learn to offload tasks to the MEC server, which reduces the probability of task failure. However, in the further training process, it can be found that there is a gap between the two algorithms. Compared with the MADDPG algorithm which does not limit the states and actions of the agents, the proposed TOCA-MADDPG algorithm can enhance the training speed, achieve convergence and get a higher reward. And when the number




(a) MADQN (b) TOCA-MADDPG (c) Difference

Fig. 3. Comparison of convergence when using different DRL method.

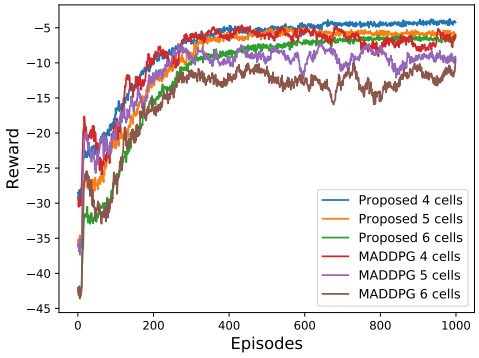
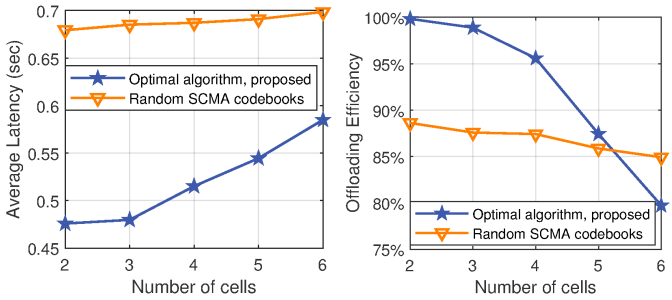



Fig. 4. The effect of state and action restrictions on convergence performance when the number of cells is large ($N > 3$).

of cells increases, the advantage becomes more obvious. In the case of four cells, there is a reward gap (2.34) between the two algorithms. When the number of cells is 5 and 6, the rewards are difficult to converge and constantly fluctuate. Compared with the average reward of MADDPG algorithm, the reward of the proposed TOCA-MADDPG algorithm increases by 37.7% (3.47) and 43.5% (4.02), respectively. Therefore, the proposed algorithm in this paper is more advantageous and can effectively train under the condition of an unstable environment.

To demonstrate the superiority of the proposed SCMA-MEC scheme, we introduce two reference schemes, which are the same as the proposed scheme except that the offloading process adopts OFDMA [33] and PD-NOMA [12] techniques, respectively. Fig. 5 shows the energy consumption and latency comparison between the proposed SCMA-MEC scheme and the multi-cell edge computing networks enabled by PD-NOMA and OFDMA. In PD-NOMA, each subcarrier can be assigned to at most two users simultaneously. It can be seen that the overall execution delay and energy consumption increase with the number of cells in all schemes. Compared with OFDMA and PD-NOMA, the energy cost and delay of the multi-cell SCMA-MEC network are significantly reduced, and the gap increases with the number of cells. In the case of two cells, the average energy consumption of SCMA is 0.064J, which is reduced by 0.072J and 0.215J in comparison to PD-NOMA and OFDMA. The average latency equals 0.478s, reduced by 0.229s and 0.347s, respectively. When the cell number is six, the average energy expenditure of SCMA is 0.267J, which is 36.6% and 47.0% lower than that of PD-NOMA and OFDMA, and the time delay comes to 0.589s, with a relative reduction of 22.4% and 34.6%.

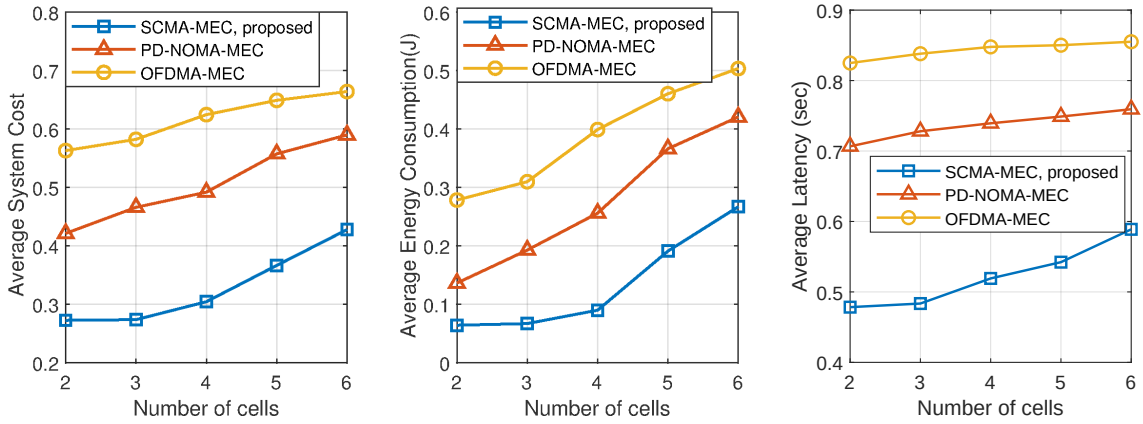



(a) Latency (b) Offloading efficiency

Fig. 6. Impact of multi-cell SCMA codebook allocation on the (a) average latency and (b) offloading efficiency, compared to random SCMA codebooks.

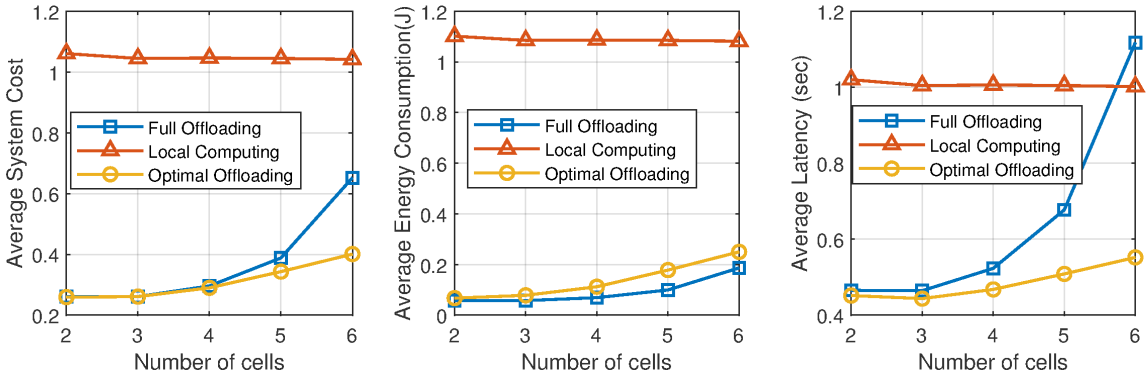
Fig. 6 depicts the impact of multi-cell SCMA codebook allocation on average system latency and offloading efficiency. The random SCMA codebook allocation uses a fixed user order algorithm [22]. We can see that the proposed algorithm has an advantage in reducing the latency compared with the random SCMA codebook allocation. In two cells, the average delay of the two schemes is 0.476s and 0.679s, respectively, and the reduction is 0.203s. In the case of six cells, the average system latency of the proposed algorithm is 0.585s, which is 0.114s lower than that of random SCMA codebook allocation. It can be deduced from the data that the proposed algorithm can reduce the latency by about 16.3%. Offloading efficiency is defined as the ratio of the number of offloaded devices to the total number of devices. Due to the inter-cell interference of multiple cells, the transmit rate decreases with the growth of cell numbers, and as a result the devices are less willing to offload tasks. Therefore, the offloading efficiency drops from 99.8% to 79.7%. Since the random codebook allocation ignores the interference between different cells and only considers the competition of multiple devices for computing resources, the offloading efficiency decreases slightly. Therefore, compared with random codebook allocation, our algorithm can achieve more channel-adapted offloading efficiency and lower system latency.

The comparison of full offloading, local computing, and near-optimal offloading is shown in Fig. 7. Overall, in terms




(a) System Cost (b) Energy (c) Latency

Fig. 5. Comparison of average (a) utility consumption, (b) energy, and (c) latency while varying the number of cells using different access schemes.




(a) System Cost (b) Energy (c) Latency

Fig. 7. Comparison of average (a) utilityz, (b) energy consumption, and (c) latency when considering full offloading, local computing and near-optimal offloading.

of average utility consumption and latency, the near-optimal offloading is smaller than that of full offloading and local computing. With a small number of cells (i.e., $2, 3, 4$), the average utility consumption of near-optimal offloading is similar to that of full offloading, suggesting that edge computing plays an important role when interference between devices is relatively low. As the number of cells gradually increases, the performance of the proposed scheme improves more significantly. With a cell count of 6, the average utility consumption of near-optimal offloading is $0.401$, which is $0.641$ $(61.5\%)$ and $0.241$ $(37.5\%)$ lower than that of local computing and full offloading, respectively, demonstrating the advantages of the proposed task offloading algorithm. In terms of energy consumption, the near-optimal offloading is similar to full offloading and much smaller than that of local computing. As the number of cells increases, the energy consumption and latency of local computing remain almost constant. This is due to the fact that devices' computing is independent of the cell count. When considering full offloading, the interference between devices increases as the number of cells increases, resulting in longer average latency. At a cell count of 2, the average local computing delay is $1.02s$, full offloading latency and near-optimal offloading delay are close to each other, which are $0.465s$ and $0.452s$ respectively. In this case, almost all of the near-optimal offloading options are to be fully offloaded to MEC servers for computation. When the number of cells is 6, the full offloading delay rises to $1.118s$ due to interference, which exceeds the local calculation delay. The near-optimal offload decision reduces the latency to $0.552s$, which is $45.9\%$ and $50.6\%$ lower than the local and full offload approaches respectively.

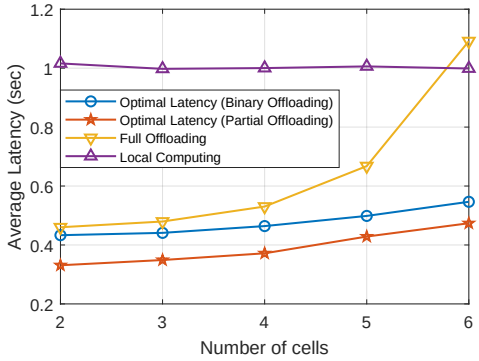
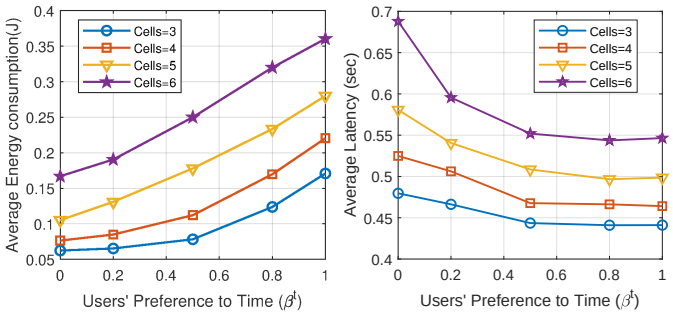



Fig. 8. Comparison of average latency versus different offloading schemes.

Fig.8 compares the differences between four computing schemes (binary offloading, partial offloading, full offloading, local computing) with the minimum latency as the optimiza-

tion objective. It can be seen that the partial offloading case outperforms the other three schemes in general. When the cell number equals 2, the near-optimal delay of binary offloading is 0.433s, which is close to full offloading. The average latency of partial offloading is 0.331s, which is 0.102s (23.6%) lower than that of binary offloading. This is because the partial offloading scheme can upload a portion of the task, which can utilize the computing power of both local devices and MEC servers to reduce task execution latency. It is also noted that the advantage of partial offloading over binary offloading decreases as the number of cells increases, with a difference of 0.073s for a cell count of 6. This is due to the fact that interference between devices reduces the percentage of devices offloaded, resulting in a lower advantage of partial offloading.




(a) Energy consumption (b) Latency

Fig. 9. Relationship between average (a) energy consumption and (b) latency with user time preference ($\beta_t$).

Fig. 9 shows the average energy consumption and time cost when changing the user's preference to time ($\beta_t$), and it is important to note that the user's preference to energy $\beta_e$ equals $1-\beta_t$. In addition, it is worth noting that when $\beta_t$ is equal to 0, the solution corresponds to the energy minimization problem, while $\beta_t = 1$, the algorithms proposed could find the near-optimal solution for the latency minimization problem. That is, this paper achieves a unified framework of energy minimization and delay minimization. It can be seen from the figure that as $\beta_t$ increases, the average delay is reduced at the cost of gradually increasing the energy consumed by devices. Furthermore, as the number of cells in the system grows, the average energy consumption and latency of the devices increases. When the optimization objective is energy minimization, the average energy consumption of the device is 0.06J in the case of three cells, and 0.167J when cells count up to 6, at an increase of 0.107J. When minimizing the latency, the average delay of the task is 0.441s when the cell number is 3, and 0.547s when the cell number equals 6, increasing by 24%. This illustrates that the device interference between multi-cells and the competition for limited resources will lead to the reduction of computation offloading benefits, resulting in an increase in energy consumption and latency.

A comparison of the average utility consumption of local computing and full offloading against the variation of task parameters is shown in Fig. 10. It can be seen from the graph that the system cost, i.e., the weighted value of energy consumption and latency increases with the growth of the required CPU computing cycles $c_u$. In addition, local computing increases

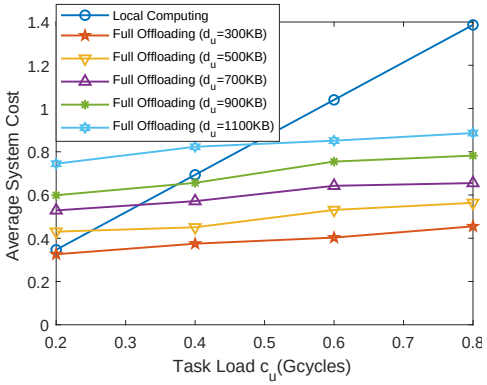
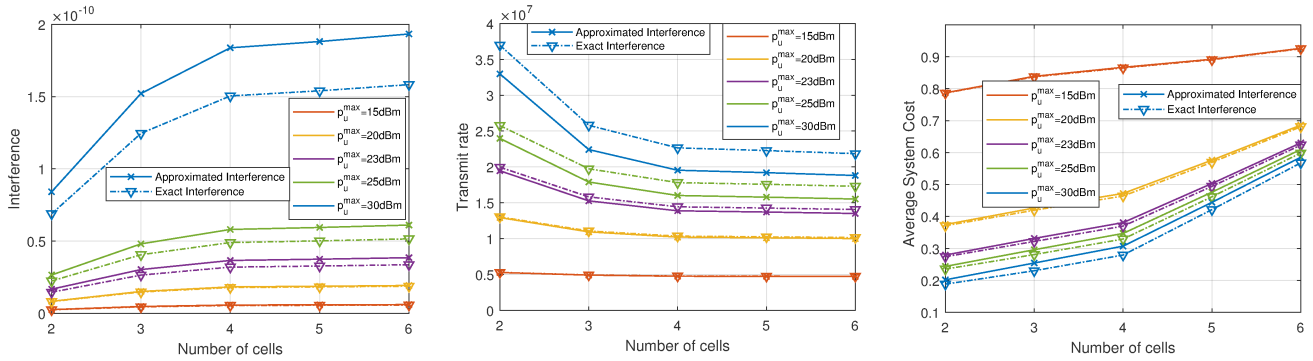



Fig. 10. Comparison of average utility consumption while varying different task parameters using different schemes.

by a larger proportion than full offloading, due to the fact that the delay and energy consumption of local computing are both related to $c_u$ and the limited local computing capacity leads to a significant increase in latency and energy consumption when $c_u$ increases. The local computing does not involve the data $d_u$ uploaded by the task, so the local utility is independent of $d_u$. The utility value of full offloading increases as $d_u$ grows. This is because the larger task input data leads to an increase in task transmit time. Specifically, when $c_u = 0.2$Gcycles and $d_u = 1100$kB, the average utility of local computing is 0.347, which owns a 53.3% gain, compared with that of full offloading (0.745). In this case, the devices prefer local computing. When $c_u$ equals 0.8Gcycles and $d_u$ equals 300kB, the average utility of full offloading equals 0.506, a 70.8% reduction compared with local computing. Therefore, it is more advantageous to assign tasks with small task description data and large computation to the MEC server.

To test and verify the effect of the inter-cell interference simplification as in (18), Fig. 11 compares the interference, transmit rate, and system cost using the approximated interference ($\tilde{I}^n_{u,c}$) vesus exact interference ($I^n_{u,c}$) under different maximum transmit power (15dBm, 20dBm, 23dBm, 25dBm, 30dBm) and the number of cells. It can be seen from Fig. 11(a) that the approximated interference is larger than the exact interference, and the gap between the two becomes smaller as the maximum transmit power of devices decreases. As the cell number grows, the inter-cell interference increases. However, the interference increases slower when the cell number is bigger, which can be explained by the smaller interference increment caused by the larger distance between devices. Fig. 11(b) describes the relationship between the transmit rate and different interferences. When the maximum transmit power is smaller, the gap of transmit rate between the approximated interference and the exact interference is smaller. With the growth of cell number, the transmit rate of SCMA decreases due to the increase of inter-cell interference. From Fig. 11(c), it can be seen that the system cost obtained by using approximate interference is almost the same as that obtained by exact interference, especially when the maximum transmit power is less than 23dBm. According to 3GPP specification [27], the maximum power of IoT devices is 23dBm, At this




(a) Interference (b) Transmit rate (c) System cost

Fig. 11. Comparison of interference, transmit rate, and system cost with approximated and exact interference under different maximum transmit power.

time, the differences between interference, transmit rate and system cost are less than 3.9% using the inter-cell interference approximation. Therefore, we can argue that the interference bound simplification is tight and the proposed algorithm is feasible in practical applications.

## V. Conclusions

In this paper, we investigated the joint resource allocation and task offloading for multi-cell SCMA-MEC systems. Aiming to minimize the system latency and devices' energy consumption, we proposed an iterative optimization scheme based on BCD for the transmit power of users and computing resources allocation of MEC servers, as well as an algorithm based on the improved simulated annealing for task offloading decision and multi-cell SCMA codebook allocation. For a dynamic environment, we propose a novel TOCA-MADDPG algorithm. Then we explained two special cases of the problem studied in this paper, namely latency minimization and energy optimization. Finally, the framework was extended to the partial offloading case, and an algorithm for solving the partial offloading ratio based on alternating convex search was proposed. Numerical results showed that the multi-cell SCMA-MEC scheme achieves lower energy consumption and system delay in comparison to the OFDMA and PD-NOMA techniques. Compared to the random codebook assignment, the multi-cell SCMA codebook allocation algorithm can achieve improved channel-adapted offloading efficiency and reduce the system latency by approximately 16.3%.

## Appendix

### A. The Proof of Lemma 1

*Proof:* Let $G_u^e(\xi_u^n, f_u^n)$ denote the objective function of problem (19). The derivatives and second-order derivatives of $G_u^e$ with respect to (w.r.t) $\xi_u^n$ and $f_u^n$ can be calculated as

$$\frac{\partial G_u^e}{\partial \xi_u^n} = \beta_t d_u + \frac{\beta_e d_u}{\zeta_{u,c}^n}\left[2^{\frac{1}{\xi_u^n}}\left(1-\frac{\ln 2}{\xi_u^n}\right)-1\right] \tag{34}$$

$$\frac{\partial G_u^e}{\partial f_u^n} = \frac{-\beta_t c_u}{\left(f_u^n\right)^2} \tag{35}$$

$$\frac{\partial^2 G_u^e}{\partial \left(\xi_u^n\right)^2} = \frac{\beta_e d_u 2^{\frac{1}{\xi_u^n}}\left(\ln 2\right)^2}{\zeta_{u,c}^n \left(\xi_u^n\right)^3} \tag{36}$$

$$\frac{\partial^2 G_u^e}{\partial \left(f_u^l\right)^2} = \frac{2\beta_t c_u}{\left(f_u^n\right)^3} \tag{37}$$

$$\frac{\partial^2 G_u^e}{\partial f_u^l \xi_u^n} = \frac{\partial^2 G_u^e}{\partial \xi_u^n f_u^l} = 0 \tag{38}$$

The values $\beta_t$, $\beta_e$, $d_u$, $\zeta_{u,c}^n$ and the variable $f_u^l$ and $\xi_u^n$ are all positive. It is easy to deduce the Hessian matrix of $G_u^e$ is a diagonal matrix with positive elements, i.e. positive-definite. Hence, the objective function is convex w.r.t $\xi_u^n$ and $f_u^n$. Additionally, the constraints (19b), (19c), (14g) and (14h) are all affine functions. Therefore, the problem (19) is proved to be a convex optimization problem. ■

### B. The Derivation of Resource Allocation in (23)

*Proof:* The Lagrangian of the problem (22) is calculated as

$$\begin{aligned}\mathcal{L}\left(f_u^n, \lambda, \mu_u\right) = &\sum_{u\in\mathcal{U}_{\text{off}}} \beta_t c_u / f_u^n + \lambda\left(\sum_{u\in\mathcal{U}_{\text{off}}} f_u^n - f_n\right) \\ &+ \sum_{u\in\mathcal{U}_{\text{off}}} \mu_u\left(\frac{c_u}{t_u^{\max} - d_u\xi_u^n} - f_u^n\right),\end{aligned} \tag{39}$$

where $\lambda$ and $\mu_u$ are non-negative Lagrange multipliers. The optimal $f_u^{n^*}$, $\lambda^*$ and $\mu_u^*$ should satisfy the following KKT equations,

$$\frac{\partial \mathcal{L}}{\partial f_u^n} = -\frac{\beta_t c_u}{\left(f_u^{n^*}\right)^2} + \lambda^* - \mu_u^* = 0, \tag{40}$$

$$f_u^{n^*} \geqslant \frac{c_u}{t_u^{\max} - d_u\xi_u^n}, \tag{41}$$

$$\sum_{u\in\mathcal{U}_{\text{off}}^n} f_u^{n^*} \leqslant f_n, \tag{42}$$

$$\lambda^* \geqslant 0, \tag{43}$$

$$\lambda^*\left(\sum_{u\in\mathcal{U}_{\text{off}}} f_u^{n^*} - f_n\right) = 0, \tag{44}$$

$$\mu_u^* \geqslant 0, \forall u \in \mathcal{U}_{\text{off}}, \tag{45}$$

$$\mu_u^*\left(\frac{c_u}{t_u^{\max} - d_u\xi_u^n} - f_u^{n^*}\right) = 0, \forall u \in \mathcal{U}_{\text{off}}. \tag{46}$$

From (40), we can get

$$f_u^{n^*} = \sqrt{\frac{\beta_t c_u}{\lambda^* - \mu_u^*}}, \forall u \in \mathcal{U}_{\text{off}}^n. \tag{47}$$

It is obvious that if $\lambda^* = 0$, the above formula (47) does not hold. Therefore, from (42), (43), and(44), the following equation can be derived:

$$\sum_{u \in \mathcal{U}_{\text{off}}} f_u^{n^*} - f_n = 0. \tag{48}$$

According to (40), (45), and (47), $f_u^*$ can be calculated w.r.t $\mu_u^*$

$$f_u^{n^*} = \begin{cases} \dfrac{c_u}{t_u^{\max} - d_u \xi_u^n} & \mu_u^* > 0 \\ \sqrt{\beta_t c_u / \lambda^*} & \mu_u^* = 0. \end{cases} \tag{49}$$

Define the set of $\mu_u = 0$ is $\mathcal{U}_0$. Let's substitute formula (49) into (48), the resource allocation solution $f_u^{n^*}$ can be deduced as

$$f_u^{n^*} = \begin{cases} \dfrac{c_u}{t_u^{\max} - d_u \xi_u^n}, & u \in \mathcal{U}_{\text{off}}^n \backslash \mathcal{U}_0 \\ \dfrac{(f_n - \sum_{u \in \mathcal{U}_{\text{off}}^n \backslash \mathcal{U}_0} f_u^{n^*}) \sqrt{c_u}}{\sum_{u \in \mathcal{U}_0} \sqrt{c_u}}, & u \in \mathcal{U}_0 \end{cases} \tag{50}$$

■

## REFERENCES


[1] Q. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020.

[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, 2016.

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

[4] 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, 2017.

[5] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach," *IEEE Internet Thing J.*, vol. 7, no. 3, pp. 1678–1689, 2020.

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

[7] S. Zarandi and H. Tabassum, "Delay minimization in sliced multi-cell mobile edge computing (MEC) systems," *IEEE Commun. Lett.*, vol. 25, no. 6, pp. 1964–1968, 2021.

[8] 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, 2019.

[9] Z. Ding, Y. Liu, J. Choi, Q. Sun, M. Elkashlan, I. Chih-Lin, and H. V. Poor, "Application of non-orthogonal multiple access in LTE and 5G networks," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 185–191, 2017.

[10] H. Nikopour and H. Baligh, "Sparse code multiple access," in *IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2013, pp. 332–336.

[11] J. V. C. Evangelista, Z. Sattar, G. Kaddoum, and A. Chaaban, "Fairness and sum-rate maximization via joint subcarrier and power allocation in uplink SCMA transmission," *IEEE Trans. Wireless Commun.*, vol. 18, no. 12, pp. 5855–5867, 2019.

[12] M. Moltafet, N. M. Yamchi, M. R. Javan, and P. Azmi, "Comparison study between PD-NOMA and SCMA," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1830–1834, 2018.

[13] Z. Ding, P. Fan, and H. V. Poor, "Impact of non-orthogonal multiple access on the offloading of mobile edge computing," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 375–390, 2019.

[14] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "Noma-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12 244–12 258, 2018.

[15] H. Li, F. Fang, and Z. Ding, "Joint resource allocation for hybrid NOMA-assisted MEC in 6G networks," *Digital Communications and Networks*, vol. 6, no. 3, pp. 241 – 252, 2020.

[16] A. Kiani and N. Ansari, "Edge computing aware NOMA for 5G networks," *IEEE Internet Thing J.*, vol. 5, no. 2, pp. 1299–1306, 2018.

[17] B. Liu, C. Liu, and M. Peng, "Joint radio and computation resource allocation for noma-enabled mec in multi-cell networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.

[18] A. Alnoman, S. Erkucuk, and A. Anpalagan, "Sparse code multiple access-based edge computing for IoT systems," *IEEE Internet Thing J.*, vol. 6, no. 4, pp. 7152–7161, 2019.

[19] P. Liu, J. Lei, and W. Liu, "An optimization scheme for SCMA-based multi-access edge computing," in *IEEE 93rd Vehicular Technology Conference (VTC-Spring)*, 2021.

[20] J. Du, W. Liu, G. Lu, J. Jiang, D. Zhai, F. R. Yu, and Z. Ding, "When mobile-edge computing (mec) meets nonorthogonal multiple access (noma) for the internet of things (IoT): System design and optimization," *IEEE Internet Thing J.*, vol. 8, no. 10, pp. 7849–7862, 2021.

[21] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core dvfs using on-chip switching regulators," in *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, 2008, pp. 123–134.

[22] M. Dabiri and H. Saeedi, "Dynamic SCMA codebook assignment methods: A comparative study," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 364–367, 2018.

[23] P. Liu, K. An, J. Lei, G. Zheng, Y. Sun, and W. Liu, "SCMA-based multiaccess edge computing in IoT systems: An energy-efficiency and latency tradeoff," *IEEE Internet Thing J.*, vol. 9, no. 7, pp. 4849–4862, 2022.

[24] K. Au, L. Zhang, H. Nikopour, E. Yi, A. Bayesteh, U. Vilaipornsawai, J. Ma, and P. Zhu, "Uplink contention based SCMA for 5G radio access," in *IEEE Globecom Workshops (GC Wkshps)*, 2014, pp. 900–905.

[25] M. B. Shahab, R. Abbas, M. Shirvanimoghaddam, and S. J. Johnson, "Grant-free non-orthogonal multiple access for IoT: A survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1805–1838, 2020.

[26] Y. Du and G. de Veciana, "wireless networks without edges: Dynamic radio resource clustering and user scheduling," in *IEEE Conference on Computer Communications (INFOCOM)*, 2014, pp. 1321–1329.

[27] 3GPP, "User Equipment (UE) radio transmission and reception," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.101, Mar 2010, version 9.3.0.

[28] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*. Springer, 1987, pp. 7–15.

[29] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6380 – 6391.

[30] W. Pan, N. Wang, C. Xu, and K.-S. Hwang, "A dynamically adaptive approach to reducing strategic interference for multi-agent systems," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–1, 2021.

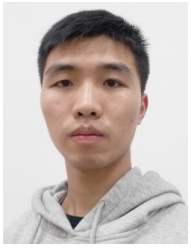
[31] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[32] M. S. Allahham, A. A. Abdellatif, N. Mhaisen, A. Mohamed, A. Erbad, and M. Guizani, "Multi-agent reinforcement learning for network selection and resource allocation in heterogeneous multi-rat networks," *IEEE Trans. Cognitive Commun. and Networking*, vol. 8, no. 2, pp. 1287–1300, 2022.

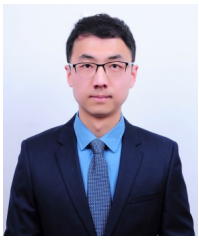
[33] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Thing J.*, vol. 5, no. 4, pp. 2633–2645, 2018.

[34] 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, 2013.

[35] B. Li, X. Deng, X. Chen, Y. Deng, and J. Yin, "Mec-based dynamic controller placement in sd-iov: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, pp. 1–15, 2022.



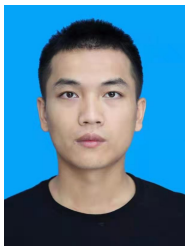
**Pengtao Liu** received the B.S. degree in communication engineering from the National University of Defence Technology (NUDT), Changsha, China, in 2019, where he is currently pursuing the Ph.D. degree with the College of Electronic Science and Technology. His current research interests include advanced multiple access techniques, multi-access edge computing, wireless resource allocation and reconfigurable intelligent surface.



**Kang An** received the B.E. degree in electronic engineering from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2011, the M.E. degree in communication engineering from PLA University of Science and Technology, Nanjing, China, in 2014, and PhD. degree in communication engineering from Army Engineering University of PLA, Nanjing, China, in 2017. Since Jan. 2018, he is with National University of Defense Technology, Nanjing, China, where he is currently a senior engineer. His current research interests are Satellite communication, B5G/6G Wireless Communication, Reconfigurable Intelligent Surface, Cognitive radio and etc.

**Jing Lei** received the B.Sc., M.Sc., and Ph.D. degrees from the National University of Defence Technology (NUDT), Changsha, China, in 1990, 1994, and 2009, respectively. She is currently a distinguished professor of Department of Communications Engineering, College of Electronic Science, National University of Defence Technology, the leader of communication coding group. She was a visiting scholar in the School of Electronics and Computer Science, University of Southampton, U,K. She has published many papers in various journals and conference proceedings and five books. Her research interests include information theory, LDPC, space-time coding, advanced multiple access technology, physical layer security, and wireless communication technology, etc.

**Wei Liu** received the B.Sc., M.Sc., and Ph.D. degrees from the National University of Defense Technology (NUDT), Changsha, China, in 2002, 2005, and 2010, respectively, where he is currently an Associate Professor of the Department of Communications Engineering, College of Electronic Science and Technology. He was a Visiting Ph.D. Student with the ECIT, Queens University of Belfast, U.K., from 2008 to 2009. His current research interests include space-time coding, OFDM, and the Internet of Things (IoT).



**Yifu Sun** received the B.Eng. in communications engineering from the National University of Defense Technology (NUDT), Changsha, China, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Electronic Science and Technology. His current research interests are in anti-jamming communications, reconfigurable intelligent surface, and physical layer security.

**Gan Zheng** (S'05-M'09-SM'12-F'21) received the BEng and the MEng from Tianjin University, Tianjin, China, in 2002 and 2004, respectively, both in Electronic and Information Engineering, and the PhD degree in Electrical and Electronic Engineering from The University of Hong Kong in 2008. He is currently Reader of Signal Processing for Wireless Communications in the Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, UK. His research interests include machine learning for communications, UAV communications, mobile edge caching, full-duplex radio, and wireless power transfer. He is the first recipient for the 2013 IEEE Signal Processing Letters Best Paper Award, and he also received 2015 GLOBECOM Best Paper Award, and 2018 IEEE Technical Committee on Green Communications & Computing Best Paper Award. He was listed as a Highly Cited Researcher by Thomson Reuters/Clarivate Analytics in 2019. He currently serves as an Associate Editor for IEEE Communications Letters and IEEE Wireless Communications Letters.

**Symeon Chatzinotas** (S'06-M'09-SM'13-F'22) received the M.Eng. degree in telecommunications from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2003, and the M.Sc. and Ph.D. degrees in electronic engineering from the University of Surrey, Surrey, U.K., in 2006 and 2009, respectively. He is currently a Full Professor/Chief Scientist and the Co-Head of the SIGCOM Research Group at SnT, University of Luxembourg. He is coordinating the research activities on communications and networking, acting as a PI for more than 20 projects and main representative for 3GPP, ETSI, DVB. He is currently serving in the editorial board of the IEEE Transactions on Communications, IEEE Open Journal of Vehicular Technology and the International Journal of Satellite Communications and Networking. In the past, he has been a Visiting Professor at the University of Parma, Italy and was involved in numerous R&D projects for NCSR Demokritos, CERTH Hellas and CCSR, University of Surrey. He was the co-recipient of the 2014 IEEE Distinguished Contributions to Satellite Communications Award and Best Paper Awards at EURASIP JWCN, CROWNCOM, ICSSC. He has (co-)authored more than 500 technical papers in refereed international journals, conferences and scientific books.