Multi-Agent Deep Reinforcement Learning for Computation Offloading and Interference Coordination in Small Cell Networks

Xiaoyan Huang, Supeng Leng, *Member, IEEE*, Sabita Maharjan, *Senior Member, IEEE*, and Yan Zhang, *Fellow, IEEE*

***Abstract*—Integrating mobile edge computing (MEC) with small cell networks has been conceived as a promising solution to provide pervasive computing services. However, the interactions among small cells due to inter-cell interference, the diverse application-specific requirements, as well as the highly dynamic wireless environment make it challenging to design an optimal computation offloading scheme. In this paper, we focus on the joint design of computation offloading and interference coordination for edge intelligence empowered small cell networks. To this end, we propose a distributed multi-agent deep reinforcement learning (DRL) scheme with the objective of minimizing the overall energy consumption while ensuring the latency requirements. Specifically, we exploit the collaboration among small cell base station (SBS) agents to adaptively adjust their strategies, considering computation offloading, channel allocation, power control, and computation resource allocation. Further, to decrease the computation complexity and signaling overhead of the training process, we design a federated DRL scheme which only requires SBS agents to share their model parameters instead of local training data. Numerical results demonstrate that our proposed schemes can significantly reduce the energy consumption and effectively guarantee the latency requirements compared with the benchmark schemes.**

***Index Terms*—Mobile edge computing, small cell networks, multi-agent deep reinforcement learning, computation offloading, interference coordination.**

I. Introduction

DRIVEN by the explosive growth of data traffic and new quality of service (QoS) requirements of mobile users, 5 G mobile networks have undergone a major shift in network architecture with the inclusion of small cell networks formed by the small cell base stations (SBSs) [1]. Meanwhile, the computation-intensive and delay-sensitive applications such as high definition video, 3-D visualization and augmented reality, define a new information-centric ubiquitous computing paradigm. However, it is challenging for the user equipments (UEs) to support those computation-intensive applications due to their limited energy and computation capacity. Mobile edge computing (MEC) emerges as a promising technology to tackle this issue, which pushes cloud services from the core network to the edge that is in close proximity to the end users [2].

Integrating MEC with small cell networks can enhance the computation and storage capabilities at the network edge. In this case, UEs can offload the computation tasks to the MEC servers at the SBSs via wireless communications, to overcome the limitations of their own computation capacity and available energy, thus supporting computation-intensive and delay-sensitive applications. However, due to the limited wireless resource and computation resource, the SBS may not be able to provide edge computing for all UEs in its coverage. Excessive offloading UEs can cause congestion in the wireless communication and the computation on MEC server, resulting in higher delay for transmission and edge computing, thus reducing the system revenue. More importantly, due to the inter-cell interference caused by the aggressive spectrum reuse scheme, the offloading performance of different cells is tightly coupled with each other. To be specific, the achievable transmission data rate for offloading in one cell depends not only on its own wireless resource allocation scheme, but also on the inter-cell interference from other cells, which is in turn determined by their wireless resource allocation schemes. Therefore, to reap the potential benefits of MEC, efficient computation offloading and interference coordination schemes are urgently needed for the multi-cell and multi-user small cell networks.

We note considerable amount of work focusing on computation offloading and resource allocation schemes based on traditional optimization methods [3]–[9]. For instance, the authors in [3] proposed a centralized computation offloading management scheme for small cell networks based on genetic algorithm and particle swarm optimization. In [4], the authors proposed a potential game-based offloading algorithm, wherein mobile devices are players that make the computation offloading decisions to minimize their own overhead. In [5], the authors proposed a two-tier greedy offloading scheme for ultra-dense networks aimed at minimizing computation overhead. The authors in [6] proposed an adaptive service offloading scheme

Manuscript received February 13, 2021; revised May 26, 2021; accepted June 23, 2021. Date of publication July 14, 2021; date of current version September 17, 2021. This work was supported in part by the National Natural Science Foundation of China under Grants 61941102 and 62071092. The review of this article was coordinated by Dr. Zhipeng Cai. *(Corresponding author: Yan Zhang.)*

Xiaoyan Huang and Supeng Leng are with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: xyhuang@uestc.edu.cn; spleng@uestc.edu.cn).

Sabita Maharjan and Yan Zhang are with the Department of Informatics, University of Oslo, 0316 Oslo, Norway, and also with the Simula Metropolitan Center for Digital Engineering, Oslo 0167, Norway (e-mail: sabita@ifi.uio.no; yanzhang@ieee.org).

Digital Object Identifier 10.1109/TVT.2021.3096928

to maximize the total revenue, while maintaining total utility value of the network. The authors in [7] proposes an edge learning-based offloading framework for autonomous driving to minimize the inference error. In [8], the authors proposed a Stackelberg game based offloading and resource allocation scheme for aerial-assisted internet of vehicles. In [9], the authors proposed a hierarchical optimization framework to optimize bandwidth allocation, offloading strategy, and relay selection sequentially, so as to reduce the latency and energy consumption. The traditional optimization methods used in literature often require complete and accurate network information, which is difficult to obtain in real networks due to highly dynamic wireless networks. Especially, the centralized optimization algorithms require global network information, imposing a large signaling burden to practical networks. Furthermore, the joint computation offloading and resource allocation problems are often modeled as combinatorial optimization problems with nonlinear constraints that are difficult to optimize efficiently using traditional optimization methods.

Edge intelligence [10] is a promising paradigm that incorporates artificial intelligence (AI) into edge networks to enable various intelligent services, such as autonomous driving [11], industrial internet of things [12], and UAV-enabled aerial surveillance [13]. Edge intelligent servers can leverage the powerful learning and reasoning ability of AI to extract valuable knowledge and make adaptive decisions, hence offering distributed and low-latency services to end users. Deep Reinforcement Learning (DRL) is an emerging technique to address the problems with stochastic and uncertain feature in complex dynamic systems [14][15]. DRL has been widely applied for optimizing computation offloading in mobile networks. For instance, the authors in [16] proposed a deep Q-learning based task offloading scheme for optimizing edge server selection and transmission mode selection, to maximize task offloading utility. In [17], the authors considered a single-MEC server network, and proposed a Deep-Q Network (DQN) based task offloading and bandwidth allocation algorithm to minimize overall offloading cost. In [18], the authors proposed a Double Deep Q-Network (DDQN) based backscatter-aided hybrid data offloading scheme to reduce power consumption in data transmission. The authors in [19] considered an end-edge-cloud orchestrated network, and proposed a Deep Deterministic Policy Gradient (DDPG) based offloading decision and computation resource allocation scheme to minimize system energy consumption. In [20], the authors proposed a DQN based computation and network resource allocation algorithm to reduce service time and balance resource utilization. In [21], the authors proposed an Asynchronous Actor-Critic (AAC) based algorithm to solve the computation offloading and resource allocation problem in a digital twin network.

The works discussed above focus on centralized intelligent approaches, which model the sophisticated global optimization problem as a single-agent reinforcement learning problem, thus requiring a central agent to collect the global state information of the environment to make the global decisions for entire system. Scalability is therefore a major issue. However, for an edge intelligence enabled small cell network, it involves interaction between multiple decentralized agents, wherein each SBS is an autonomous agent, able to make its decision individually based on its local observation of the environment. In such multi-agent environment, it's of critical importance to design a decentralized intelligent algorithm to coordinate the actions of the SBS agents to improve the overall performance of the system. Multi-agent reinforcement learning (MARL) [22] is a promising distributed machine learning approach, particularly suitable for the multi-agent network scenarios where each agent only has local information about the global environment. In the literature, MARL has been employed to address different network problems, such as dynamic power allocation in wireless networks [23], spectrum allocation for D2D underlay communications [24], and resource allocation for UAV networks [25].

In this paper, we focus on distributed intelligent computation offloading and interference coordination scheme for small cell networks. In particular, we investigate the joint optimization of computation offloading, channel allocation, power control, and computation resource allocation for multi-user and multi-cell networks based on MARL. The key contributions of our work are summarized as follows:

- We formulate a joint computation offloading and interference coordination problem for an edge intelligence enabled small cell network as a multi-agent DRL problem to minimize the system energy consumption taking stringent delay constraints into account.
- We propose a distributed multi-agent DRL-based computation offloading and resource allocation algorithm, wherein the SBS agents independently take actions based on their local observations, but refine their strategies through collaborative exploration of the environment, so as to coordinate the inter-cell interference and improve the overall system performance.
- To decrease the computing complexity and signaling overhead of the training process, we propose a federated DRL-based computation offloading and resource allocation algorithm, which only requires SBS agents to share their model parameters instead of local training data.

The remainder of this paper is organized as follows. The system model of the edge intelligence enabled small cell network is presented in Section II. The proposed multi-agent DRL and federated DRL based computation offloading and resource allocation schemes are introduced in Section III and Section IV, respectively. Numerical results are presented in Section V. Finally, the paper is concluded in Section VI.

II. System Model

In this section, we first introduce the edge intelligence enabled small cell network model, then present the communication model and the computation model in detail.

A. Edge Intelligence Enabled Small Cell Network

We consider an edge intelligence enabled small cell network, consisting of M randomly distributed SBSs and N randomly distributed UEs, as shown in Fig. 1. Both SBSs and UEs are equipped with a single antenna. The SBSs are equipped with MEC servers and capable of providing artificial intelligence enhanced decision capabilities at the network edge. Denote the

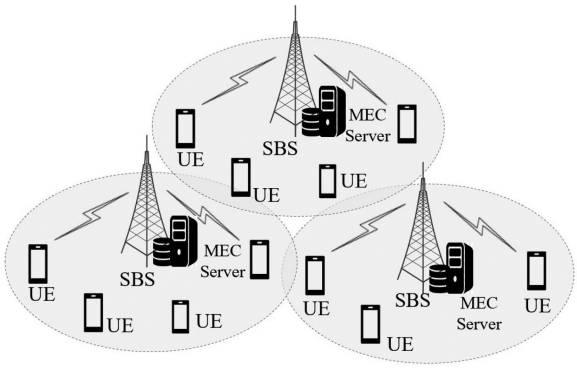

Fig. 1. Edge intelligence enabled small cell network

set of SBSs as $\mathcal{M} = \{0, 1, 2, \ldots, M\}$ and the set of UEs served by SBS $m \in \mathcal{M}$ as $\mathcal{N}_m = \{1, 2, \ldots, N_m\}$ with $\mathcal{N}_m \cap \mathcal{N}_{m'} = \emptyset$ for $m \neq m'$, and $\sum_m \mid \mathcal{N}_m \mid = N$.

Each UE has a computation intensive task to complete. The computation task of UE n in cell m can be described as $Task_n^{(m)} = \{d_n^{(m)}, \omega_n^{(m)}, \tilde{T}_n^{(m)}\}$, where $d_n^{(m)}$ denotes the size of the computation task, and $\omega_n^{(m)}$ represents the total computation capability (i.e., CPU cycles) required to complete the computation task, and $\tilde{T}_n^{(m)}$ is the maximum latency that can be tolerated by the UE. To complete the computation task, the UEs can compute locally, or offload the computation task to the corresponding serving SBSs. Differentiated from the binary offloading scheme in [3] and [4], we consider the partial offloading scheme to improve the granularity of offloading, wherein a part of a computation task can be executed locally at the UE, and the remaining part is executed remotely at the MEC server of the serving SBS at the same time, i.e., the local computing and edge computing respectively.

When a UE decides to offload, the offloaded part of its task will be transmitted to the MEC server of its serving SBS through wireless uplink. Then the MEC server of the serving SBS executes the computation task for the UE, and the serving SBS sends the computation results back to the UE through wireless downlink. As in [3] and [19], we focus on the first two phases, involving a communication model and a computation model, which will be presented in detail in the following subsections.

B. Communication Model

Define $\alpha_n^{(m)} \in [0, 1]$ as the offloading ratio of UE n in cell m. Accordingly, for a computation task with size of $d_n^{(m)}$, the part with size $\alpha_n^{(m)} d_n^{(m)}$ is executed at the edge server, and the other part with size $(1 - \alpha_n^{(m)}) d_n^{(m)}$ is executed locally. Especially, $\alpha_n^{(m)} = 0$ represents that the UE decides to complete the whole computation task locally by itself, and $\alpha_n^{(m)} = 1$ means that the UE decides to offload the whole computation task to the MEC server of its serving SBS. When $0 < \alpha_n^{(m)} < 1$, a part of the computation task with size $\alpha_n^{(m)} d_n^{(m)}$ will be offloaded to its serving SBS through wireless uplink.

Let K denote the number of orthogonal channels in the system, and SBSs reuse the full spectrum of the system. In this case, there exists co-channel interference among different small cells. In each small cell, the orthogonal frequency-division multiple access scheme is adopted, so that there is no intra-cell interference. Define $b_{n,k}^{(m)} \in \{0, 1\}$ as the channel allocation indicator for SBS m, where $b_{n,k}^{(m)} = 1$ represents that SBS m allocates channel k to UE n, and otherwise $b_{n,k}^{(m)} = 0$. Considering the orthogonal frequency-division multiple access scheme adopted in each small cell, each channel can be assigned to at most one UE. Thus the channel allocation decisions of the SBSs must satisfy the constraint $\sum_{n \in \mathcal{N}_m} b_{n,k}^{(m)} \leq 1, \forall k \in \mathcal{K}, m \in \mathcal{M}$. Furthermore, we assume that each UE can be allocated at most one channel when offloading computation task to the serving SBS, that is $\sum_{k \in \mathcal{K}} b_{n,k}^{(m)} \leq 1, \forall n \in \mathcal{N}_m, m \in \mathcal{M}$.

Define $p_n^{(m)} \in [0, P_n^{(m),max}]$ as the transmit power of UE n in cell m, where $P_n^{(m),max}$ is the maximum transmit power of the UE. Hence, when UE n in cell m offloads the computation task to its serving SBS via channel k, the achievable uplink data rate on channel k for UE n is given by

$$r_{n,k}^{(m)} = B \log_2 \left(1 + \frac{p_n^{(m)} h_{n,k}^{(m)}}{N_0 B + I_k^{(m)}} \right) \tag{1}$$

where B is the bandwidth of each channel. $h_{n,k}^{(m)}$ is the channel gain on channel k for the signal link from UE n to its serving SBS m. N_0 is the spectrum density of the additive white Gaussian noise (AWGN). $I_k^{(m)} = \sum_{m' \neq m} \sum_{n' \in \mathcal{N}_{m'}} p_{n'}^{(m')} g_{n',k}^{(m)}$ is the received interference on channel k for SBS m, where $g_{n',k}^{(m)}$ is the channel gain on channel k for the interfering link from UE n' to SBS m. Therefore, the achievable uplink data rate of UE n in cell m is given by

$$U_n^{(m)} = \sum_{k \in \mathcal{K}} b_{n,k}^{(m)} r_{n,k}^{(m)}. \tag{2}$$

According to (1) and (2), the achievable uplink data rate of a UE for offloading depends not only on the allocated channel and its own transmit power, but also on the inter-cell interference from the offloading UEs in other cells. From the perspective of system optimization, the coordination among different small cells for computation offloading, channel allocation, and power control is vital to alleviating the inter-cell interference and hence improving the offloading performance.

C. Computation Model

Given the offloading ratio $\alpha_n^{(m)}$ of UE n in cell m, for the total computation capability requirement $\omega_n^{(m)}$, the part $\alpha_n^{(m)} \omega_n^{(m)}$ is executed through the edge computing, and the other part $(1 - \alpha_n^{(m)}) \omega_n^{(m)}$ is executed by the local computing.

1) Local Computing: Assume that the computation capacity (i.e., CPU frequency) of each UE is fixed, but may vary over the UEs. Denote the CPU frequency of UE n in cell m as $C_n^{(m)}$. Thus, given the offloading ratio $a_n^{(m)}$, the delay for local computing can be given by

$$LocT_n^{(m)} = (1 - \alpha_n^{(m)}) \omega_n^{(m)} / C_n^{(m)}. \tag{3}$$

Furthermore, given the energy consumption coefficient ${\varepsilon_n}^{(m)}$ for per CPU cycle of UE n in cell m, the energy consumption for local computing is given by

$$LocE_n^{(m)} = (1 - {\alpha_n}^{(m)})\omega_n^{(m)}{\varepsilon_n}^{(m)}. \tag{4}$$

2) Edge Computing: According to the communication model presented in section II-B, given the offloading ratio indicator $\alpha_n^{(m)}$ of UE n in cell m, the delay and energy consumption for transmission to SBS m are respectively given by

$$OffT_n^{(m),tr} = \alpha_n^{(m)} d_n^{(m)} / U_n^{(m)} \tag{5}$$

and

$$OffE_n^{(m),tr} = p_n^{(m)} \alpha_n^{(m)} d_n^{(m)} / U_n^{(m)}. \tag{6}$$

Denote the computation capacity of the MEC server at SBS m as $\bar{C}^{(m)}$, which is finite and fixed, but may vary over the SBSs. For UE $n \in \mathcal{N}_m$ offloading to SBS m, define $f_n^{(m)} \in (0, 1]$ as the computation resource assignment ratio for UE n. Considering the finite computation capacity of the MEC server, the computation resource assigned to the UEs cannot exceed the total available computation resource in the MEC server, that is $\sum_{n \in \mathcal{N}_m} f_n^{(m)} \leq 1$, $\forall m \in \mathcal{M}$. Accordingly, the delay of UE n for edge computing can be expressed as

$$OffT_n^{(m),ex} = \alpha_n^{(m)} \omega_n^{(m)} / f_n^{(m)} \bar{C}^{(m)}. \tag{7}$$

Furthermore, given the energy consumption coefficient $\bar{\varepsilon}^{(m)}$ per CPU cycle of the MEC servers at SBS m, the energy consumption of UE n for edge computing is given by

$$OffE_n^{(m),ex} = \alpha_n^{(m)} \omega_n^{(m)} \bar{\varepsilon}^{(m)}. \tag{8}$$

Therefore, the delay and energy consumption of UE n for offloading the task to the serving SBS can be respectively written as

$$OffT_n^{(m)} = OffT_n^{(m),tr} + OffT_n^{(m),ex} \tag{9}$$

and

$$OffE_n^{(m)} = OffE_n^{(m),tr} + OffE_n^{(m),ex}. \tag{10}$$

III. Multi-Agent DRL for Computation Offloading and Resource Allocation

According to the communication model and computation model presented in section II-B and section II-C, the offloading performance of different small cells is coupled with each other due to the inter-cell interference. Considering the limited communication and computation resources in each small cell, how to coordinate the computation offloading and resource allocation schemes of the small cells to improve the overall performance of the system remains challenging. Consequently, the problem of interest is to jointly optimize the computation offloading decision, channel allocation, power control, and computation resource allocation of the multiple small cells, with the objective of minimizing the total energy consumption while satisfying the latency requirements of the computation tasks. We consider a fully synchronized time slotted system, with each time slot as a scheduling period. The computation offloading and resource allocation problem in slot t is formulated as follows:

$$\begin{aligned}
&\text{P0:} \min_{\boldsymbol{\alpha}[t], \mathbf{b}[t], \mathbf{p}[t], \mathbf{f}[t]} \sum_m \sum_n \Big(LocE_n^{(m)}[t] + OffE_n^{(m)}[t] \Big) \\
&s.t. \\
&\text{C1:} \max\{LocT_n^{(m)}[t], OffT_n^{(m)}[t]\} \leqslant \tilde{T}_n^{(m)}, \forall m, n \\
&\text{C2:} \ \alpha_n^{(m)}[t] \in [0, 1], \forall n, m \\
&\text{C3:} \ b_{n,k}^{(m)}[t] \in \{0, 1\}, \forall n, m, k \\
&\text{C4:} \sum_{k \in \mathcal{K}} b_{n,k}^{(m)}[t] \leq 1, \forall n, m \\
&\text{C5:} \sum_{n \in \mathcal{N}_m} b_{n,k}^{(m)}[t] \leq 1, \forall k, m \\
&\text{C6:} \ p_n^{(m)}[t] \in [0, P_n^{(m),max}], \forall m, n \\
&\text{C7:} \ f_n^{(m)}[t] \in (0, 1], \sum_{n \in \mathcal{N}_m} f_n^{(m)}[t] \leq 1, \forall m
\end{aligned} \tag{11}$$

where $\boldsymbol{\alpha}[t] = \{\alpha_n^{(m)}[t]\}$, $\mathbf{b}[t] = \{b_{n,k}^{(m)}[t]\}$, $\mathbf{p}[t] = \{p_n^{(m)}[t]\}$, and $\mathbf{f}[t] = \{f_n^{(m)}[t]\}$. With the partial offloading scheme, the delay of completing a computation task depends on the longer one between the time for local computing and edge computing. Therefore, constraint C1 is considered to ensure the latency requirements of the computation tasks. Constraint C2 states that the fractional offloading scheme is supported in this work. Constraints C3-C5 represent that each UE can be allocated with at most one channel, and each channel in each cell can be assigned to at most one UE. Constraint C6 represents the transmit power constraint of each UE. Constraint C7 means that the amount of the computation resource allocated to all UEs cannot exceed the total available computation resource in the MEC server in each cell.

The global optimization problem P0 in (11) is a mixed integer non-linear programming problem (MINLP), wherein the channel allocation indicator $b_{n,k}^{(m)}[t]$ is binary variable, while the offloading ratio $\alpha_n^{(m)}[t]$, the transmit power $p_n^{(m)}[t]$, and the computation resource assignment ratio $f_n^{(m)}[t]$ are real positive numbers. In addition, the problem P0 is non-convex due to the inter-cell interference terms in the achievable data rate in (1), and combinatorial due to the binary variable, such that it can not be solved directly even for a small-scale network. In the practical small cell networks, the number of UEs and SBSs are increasing over time, resulting in the significant increase on the complexity of our problem. DRL has been recognized as an efficient method to find an optimal policy in complex dynamic systems. Therefore, we attempt to exploit DRL to tackle the challenge of solving the global optimization problem in this work.

A. Multi-Agent Environment Modeling

In the edge intelligence enabled small cell network illustrated in Fig. 1, the SBS of each small cell decides the computation

offloading and resource allocation scheme for the UEs it serves according to its local environment and user demands in each time slot. Due to the inter-cell interference, the decisions of different SBSs affect each other, resulting in tight coupling of the performance of different small cells. Therefore, it can be modelled as a multi-agent reinforcement learning problem, where each SBS acts as an agent and interacts with the environment to gain experiences to improve its policy of computation offloading and resource allocation.

In such multi-agent environment, the SBS agents independently update their policies as learning progresses. It's noteworthy to mention that if two or more agents update simultaneously, the environment appears non-stationary from the perspective of any individual agent, resulting in instability in the training process. As in [23], [24], and [25], the considered multi-agent reinforcement learning problem can be modelled as a partially observable Markov game [26], which is the generalization of the Markov decision processes to the multi-agent case. Specifically, at each time t, given the current environment state $S[t]$, each SBS agent m receives an observation $s_m[t] = O(S[t], m)$ of the environment with the observation function O, and then takes an action $a_m[t]$, forming a joint action $\mathbf{a}[t] = \{a_1[t], \ldots, a_M[t]\}$ of all agents. Thereafter, the agent receives the immediate reward $R_m[t]$ based on the joint action $\mathbf{a}[t]$, and the environment evolves to the next state $S[t+1]$. The new observation $s_m[t+1]$ is then received by the SBS agent m with a transition probability of $p(s_m[t+1]|s_m[t], a_1[t], \ldots, a_M[t])$. In our system, the state space, the action space, and the reward function are defined as follows.

1) State Space: The environment state $S[t]$ may include the global channel condition and the behaviours of all SBS agents. In real networks, it is not practical to assume that each SBS agent has knowledge about the global environment state. Instead, each SBS can acquire partial knowledge of the environment through an observation function. The state observed by SBS agent m at time t for characterizing the environment contains the local instant channel gains of its own signal links on each channel, the received interference power at SBS m on each channel, as well as the task profiles of the UEs in its coverage. Accordingly, the local observation of SBS agent m can be summarized as $s_m[t] = \{\mathbf{h}_m[t], \mathbf{I}_m[t], \mathbf{Task}_m[t]\}$, where

- $\mathbf{h}_m[t] = \{[h_{1,1}^{(m)}[t], \ldots, h_{1,K}^{(m)}[t]], \ldots\ldots, [h_{|\mathcal{N}_m|,1}^{(m)}[t], \ldots, h_{|\mathcal{N}_m|,K}^{(m)}[t]]\}$ represents the instant channel gains of the signal links between UE $n \in \mathcal{N}_m$ and SBS m on channel $k \in \mathcal{K}$, which can be accurately estimated by SBS m.
- $\mathbf{I}_m[t] = [I_1^{(m)}[t], \ldots, I_K^{(m)}[t]]$ represents the received interference power at SBS m, which can be measured separately on each channel $k \in \mathcal{K}$ at SBS agent m.
- $\mathbf{Task}_m[t] = [Task_1^{(m)}[t], \ldots, Task_{|\mathcal{N}_m|}^{(m)}[t]]$ represents the task profiles of the UEs served by SBS m.

2) Action Space: At each time t, SBS agent m takes an action $a_m[t]$ according to the current local observation $s_m[t]$ based on its decision policy. The computation offloading and resource allocation design of SBS m comes down to the computation offloading decision, channel allocation, uplink power control, and computation resource allocation for the UEs served by SBS m. Accordingly, the action of SBS agent m can be defined as $a_m[t] = \{\boldsymbol{\alpha}_m[t], \mathbf{b}_m[t], \mathbf{p}_m[t], \mathbf{f}_m[t]\}$, where

- $\boldsymbol{\alpha}_m[t] = [\alpha_1^{(m)}[t], \ldots, \alpha_{|\mathcal{N}_m|}^{(m)}[t]]$ denotes offloading decisions for each UE $n \in \mathcal{N}_m$, i.e., the ratio of task executed by the edge computing.
- $\mathbf{b}_m[t] = \{[b_{1,1}^{(m)}[t], \ldots, b_{1,K}^{(m)}[t]], \ldots\ldots, [b_{|\mathcal{N}_m|,1}^{(m)}[t], \ldots, b_{|\mathcal{N}_m|,K}^{(m)}[t]]\}$ denotes the channel assigned to each UE $n \in \mathcal{N}_m$ for transmitting the offloaded task to SBS m.
- $\mathbf{p}_m[t] = [p_1^{(m)}[t], \ldots, p_{|\mathcal{N}_m|}^{(m)}[t]]$ denotes the uplink transmit power of each UE $n \in \mathcal{N}_m$ for offloading task to the serving SBS m. Note that $p_n^{(m)}[t] = 0$ when UE n computes its task locally.
- $\mathbf{f}_m[t] = [f_1^{(m)}[t], \ldots, f_{|\mathcal{N}_m|}^{(m)}[t]]$ denotes the computation resource that SBS m allocates to each UE $n \in \mathcal{N}_m$. And $f_n^{(m)}[t] = 0$ when UE n computes its task locally.

3) Reward Function: Each SBS agent refines its computation offloading and resource allocation policy through a learning process driven by the reward function, where the reward function design correlates with the desired objective. According to the optimization problem P0 in (11), we consider a cooperative multi-agent scenario, where the SBS agents should cooperate with each other rather than acting selfishly in their own interests to compromise the overall performance of the system. To this end, a system performance-oriented reward function is required to promote cooperation among SBS agents, such that the SBS agents improve their policies toward achieving the desired global objective. Since our goal is to minimize the total energy consumption while satisfying the latency requirements of the computation tasks, the immediate reward $R_m^{Co}[t]$ for SBS agent m at time t can be defined as

$$R_m^{Co}[t] = \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}_m} \left(-LocE_n^{(m)}[t] - OffE_n^{(m)}[t] + V_n^{(m)}[t] \right) \tag{12}$$

where $LocE_n^{(m)}[t]$ and $OffE_n^{(m)}[t]$ denote the energy consumption for executing task $Task_n^{(m)}$ locally and remotely at SBS m, respectively, at time t. Moreover, $V_n^{(m)}[t]$ is defined as

$$V_n^{(m)}[t] = G\left(\tilde{T}_n^{(m)} - \max\left\{ LocT_n^{(m)}[t], OffT_n^{(m)}[t] \right\} \right) \tag{13}$$

where $G(x)$ is a piecewise function, i.e., $G(x) = \beta$ if $x \geq 0$, and otherwise $G(x) = x$ if $x < 0$. The immediate reward (12) is composed of three parts. The first and second parts correspond to the energy consumed by local computing and edge computing, respectively. The third part denotes the impact of the latency requirements of the computation tasks. According to (13), $V_n^{(m)}[t]$ is set to a positive constant number, β, representing the revenue when the latency requirement of the computation task is guaranteed, and otherwise is set to a negative penalty, with an absolute value equal to the difference between the latency requirement and the actual delay of executing the computation task. In practice, β is a hyperparameter, which is tuned empirically such that it is greater than the absolute of the penalty, but not

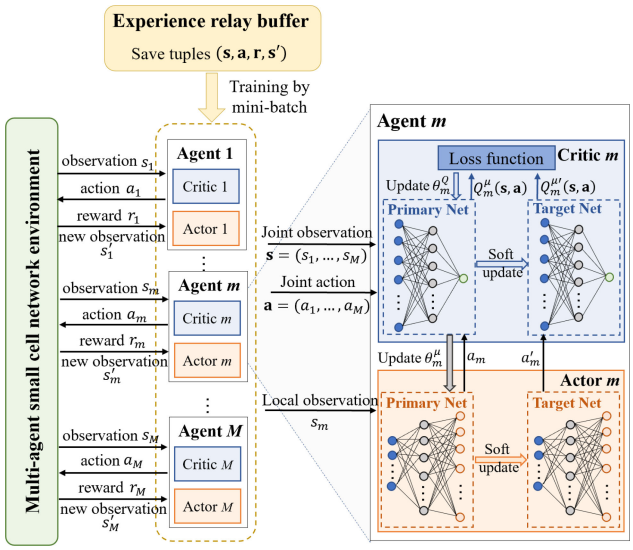

Fig. 2. The multi-agent DRL scheme

obscuring the impact of the first two parts in the reward function, i.e., the total energy consumption. Accordingly, the learning process attempts to reduce the overall energy consumption and fulfill the latency requirements of as many computation tasks as possible to maximize the expected cumulative rewards.

The goal of learning for agent m is to find a joint computation offloading and resource allocation policy to maximize the expected cumulative discounted reward, i.e.,

$$J_m = \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t R_m^{Co}[t]\right) \tag{14}$$

where the constant $\gamma \in [0, 1)$ is the discount factor.

B. Multi-Agent DRL Based Computation Offloading and Resource Allocation Scheme

In order to improve the overall performance of the system and overcome the inherent nonstationarity of the multi-agent environment, based on the multi-agent deep deterministic policy gradient (MADDPG) [27], we propose a multi-agent DRL based computation offloading and resource allocation scheme for small cell networks. Different from the existing DRL based centralized algorithms proposed in [16]–[21], the proposed multi-agent DRL based algorithm is a distributed approach, alleviating the scalability issue and more suitable to the practical networks. In the proposed multi-agent DRL based algorithm, the SBS agents independently take actions based on their local observations and user demands, but refine their strategies of computation offloading, channel allocation, power control, and computation resource allocation through collaborative exploration of the environment, so as to coordinate the inter-cell interference thus improving the overall energy consumption performance of the system.

As shown in Fig. 2, each SBS is modelled as a DDPG agent, consisting of two parts, i.e., the actor and the critic networks. The input of the actor network is the local state observed by the agent, and the output is its selected action. However, besides the local observation and action of the agent itself, the input of the critic network is also augmented by extra information about local observations and actions of the other agents. The output of the critic network is the corresponding Q-value. Denote the set of actor networks and critic networks of all SBS agents as $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M\}$ and $\mathbf{Q} = \{Q_1, \ldots, Q_M\}$ parameterized by $\boldsymbol{\theta}^{\boldsymbol{\mu}} = \{\theta_1^{\boldsymbol{\mu}}, \ldots, \theta_M^{\boldsymbol{\mu}}\}$ and $\boldsymbol{\theta}^{Q} = \{\theta_1^{Q}, \ldots, \theta_M^{Q}\}$, respectively. Meanwhile, in order to overcome the divergence of Q-value update, a copy of the primary actor and critic networks are also created for each agent, i.e., the target actor network $\boldsymbol{\mu}'_m$ and target critic network Q'_m for SBS agent m with delayed weight $\theta_m^{\boldsymbol{\mu}'}$ and $\theta_m^{Q'}$, respectively.

To guarantee the non-correlation in training data, an experience replay buffer is utilized to store the transition tuples $(\mathbf{s}[t], \mathbf{a}[t], \mathbf{r}[t], \mathbf{s}[t+1])$, where $\mathbf{s}[t] = \{s_1[t], \ldots, s_M[t]\}$ is the joint observation, $\mathbf{a}[t] = \{a_1[t], \ldots, a_M[t]\}$ is the joint action, and $\mathbf{r}[t] = \{r_1[t], \ldots, r_M[t]\}$ is the joint reward with $r_m[t] \triangleq R_m^{Co}[t]$ given by (12). At each time step, the critic and actor networks can be updated by sampling a minibatch uniformly from the experience replay buffer. To be specific, the critic network of SBS agent m is updated by minimizing the loss function $L_m(\theta_m^Q)$, which is defined as

$$L_m(\theta_m^Q) = \frac{1}{W}\sum_{j=1}^{W}\left[y_m^j - Q_m^{\boldsymbol{\mu}}\left(\mathbf{s}^j, a_1^j, \ldots, a_M^j\right)\right]^2 \tag{15}$$

where W is the size of mini-batches, and j is the index of the randomly sampled mini-batches. y_m^j is the target value calculated by the target critic network, given by

$$y_m^j = r_m^j + \gamma Q_m^{\boldsymbol{\mu}'}(\mathbf{s}'^j, a'_1, \ldots, a'_M)\,|_{a'_k = \boldsymbol{\mu}'_k(s_k^j)}\,. \tag{16}$$

Different from the action-value function in the single-agent DRL algorithm which takes as input only the local observation and action, the action-value function $Q_m^{\boldsymbol{\mu}}(\mathbf{s}, \mathbf{a})$ of SBS agent m in the proposed multi-agent collaborative DRL algorithm takes as input the joint observation $\mathbf{s}$ and joint action $\mathbf{a}$. The joint observation $\mathbf{s}$ and joint action $\mathbf{a}$ consist of the observations and actions of all SBS agents, respectively. As such, the critic network of each agent can evaluate the quality of its selected action taking the actions of other agents into consideration, thereby promoting the coordination among the agents.

Then, based on (15) and (16), the weight θ_m^Q of the critic network can be updated by

$$\theta_m^Q \leftarrow \theta_m^Q - \delta \nabla_{\theta_m^Q} L(\theta_m^Q) \tag{17}$$

where δ is the learning rate.

Moreover, the actor network of SBS agent m can be updated with the policy gradient scheme as follows.

$$\nabla_{\theta_m^{\boldsymbol{\mu}}} J_m(\theta_m^{\boldsymbol{\mu}}) =$$

$$\frac{1}{W}\sum_{j=1}^{W} \nabla_{\theta_m^{\boldsymbol{\mu}}} \boldsymbol{\mu}(s_m^j) \nabla_{a_m} Q_m^{\boldsymbol{\mu}}(\mathbf{s}^j, a_1^j, ., a_m, ., a_M^j)|_{a_m = \boldsymbol{\mu}(s_m^j)} \tag{18}$$

Subsequently, the corresponding weight $\theta_m^{\boldsymbol{\mu}}$ can be updated by

$$\theta_m^{\boldsymbol{\mu}} \leftarrow \theta_m^{\boldsymbol{\mu}} - \delta \nabla_{\theta_m^{\boldsymbol{\mu}}} J(\theta_m^{\boldsymbol{\mu}}). \tag{19}$$

On the other hand, the target network can be regarded as an old version of the primary network with delayed weights. With the soft updating strategy, the weights $\theta_m^{Q'}$ and $\theta_m^{\boldsymbol{\mu}'}$ of the target critic and target actor networks of SBS agent m can be updated by

$$\begin{aligned} \theta_m^{Q'} &= \tau\theta_m^{Q} + (1-\tau)\theta_m^{Q'} \\ \theta_m^{\boldsymbol{\mu}'} &= \tau\theta_m^{\boldsymbol{\mu}} + (1-\tau)\theta_m^{\boldsymbol{\mu}'} \end{aligned} \tag{20}$$

where τ is the soft updating rate of the target network.

C. Algorithm for Multi-Agent DRL

The proposed multi-agent DRL based computation offloading and resource allocation algorithm is summarized as Algorithm 1. In implementation, the proposed multi-agent DRL based computation offloading and resource allocation algorithm is divided into the training and execution phases according to the centralized learning and distributed implementation framework.

In the training phase, the historical global information about the observations, actions, and rewards of all SBS agents is utilized to train the actor and critic networks of each SBS agent with the experience replay strategy, so that the SBS agents collaboratively improve their policies for the global objective. Specifically, the training procedure consists of multiple episodes, each consisting of multiple time steps. In each time step, each SBS agent decides its actions $a_m[t]$ in computation offloading, channel allocation, power control, and computation resource allocation by the primary actor network $\boldsymbol{\mu}_m(s_m[t])$ based on its current policy and local observation. Then, each SBS informs its UEs the decisions on computation offloading, channel allocation, and power control via an intra-cell dedicated control channel. Meanwhile, all SBS agents broadcast their local observations and selected actions via an inter-cell dedicated control channel, so that each SBS agent can compute the immediate reward $r_m[t]$ according to (12). Based on the actions taken by all SBS agents, the environment transits to a new state due to the changes of co-channel interference and small-scale channel fading. Each SBS agent acquires its next local observation $s_m[t+1]$ subsequently. Thereafter, based on mini-batch technique, each SBS agent updates its critic and actor networks according to (15)–(20).

In the execution phase, each SBS agent individually makes decisions on computation offloading, channel allocation, power control, and computation resource allocation by its trained actor network based on its local observation of the entire environment. Then, each SBS informs its UEs the decisions on computation offloading, channel allocation, and power control via an intra-cell dedicated control channel, then the UEs can perform accordingly.

The time complexity of Algorithm 1 mainly depends on the number of the SBS agents and the structure of the neural networks for implementing the actor and critic networks of each SBS agent. Assuming that the actor network and critic network of each SBS agent contains J and L fully connected layers, respectively, the time complexity can be calculated as

$$\begin{aligned} &M \times \left(2 \times \sum_{j=0}^{J} n_{A,j} n_{A,j+1} + 2 \times \sum_{l=0}^{L} n_{C,l} n_{C,l+1} \right) \\ &\quad = \mathcal{O}\left(M \times \left(\sum_{j=0}^{J} n_{A,j} n_{A,j+1} + \sum_{l=0}^{L} n_{C,l} n_{C,l+1} \right) \right) \end{aligned} \tag{21}$$

where $n_{A,j}$ and $n_{C,l}$ represent the unit number in j-th actor net layer and l-th critic net layer, respectively. M is the number of the SBS agents in the system. $n_{A,0}$ and $n_{C,0}$ equal the input size of the correspond network.

Algorithm 1: Multi-Agent DRL Based Computation Offloading and Resource Allocation Algorithm.

1: Initialize actor networks $\boldsymbol{\mu}$ and critic networks $\mathbf{Q}$ with weights $\boldsymbol{\theta}^{\boldsymbol{\mu}}$ and $\boldsymbol{\theta}^{Q}$.
2: Initialize target actor networks $\boldsymbol{\mu}'$ and target critic networks $\mathbf{Q}'$ with weights $\boldsymbol{\theta}^{\boldsymbol{\mu}'}$ and $\boldsymbol{\theta}^{Q'}$.
3: Initialize the experience replay buffer $\mathcal{D}$;
4: **for** each episode $epi = 1, 2, \ldots$ **do**
5: Initialize the local observation state $\mathbf{s}[0]$;
6: **for** each time $t = 1, 2, \ldots,$ **do**
7: All SBS agents select actions $\mathbf{a}[t] = \{a_m[t] = \boldsymbol{\mu}_m(s_m[t]), m \in \mathcal{M}\}$, and execute the selected actions;
8: All SBS agents compute the reward $\mathbf{r}[t] = \{r_m[t] = R_m^{Co}[t], m \in \mathcal{M}\}$ with (12), and observe the next state $\mathbf{s}[t+1]$;
9: Save the tuple $(\mathbf{s}[t], \mathbf{a}[t], \mathbf{r}[t], \mathbf{s}[t+1])$ in $\mathcal{D}$;
10: Sample a random mini-batch of tuples from $\mathcal{D}$;
11: Update the critic networks of all SBS agents with (15), (16) and (17);
12: Update the actor networks of all SBS agents with (19) and (18);
13: Update the target actor and target critic networks with (20);
14: **end for**
15: **end for**

IV. Federated DRL for Computation Offloading and Resource Allocation

In the proposed multi-agent DRL algorithm, the global information about the observations and actions of all SBS agents is required to train the actor and critic networks of each SBS agent. Therefore, the SBS agents need to exchange their local information with each other, which may lead to significant signaling overhead. In this section, we will design a low-overhead algorithm for joint computation offloading and resource allocation, which is more preferred for the resource-constrained scenarios.

Recall that problem P0 in (11) is a global optimization problem to minimize the overall energy consumption of the system. Since the different small cells are tightly coupled through the

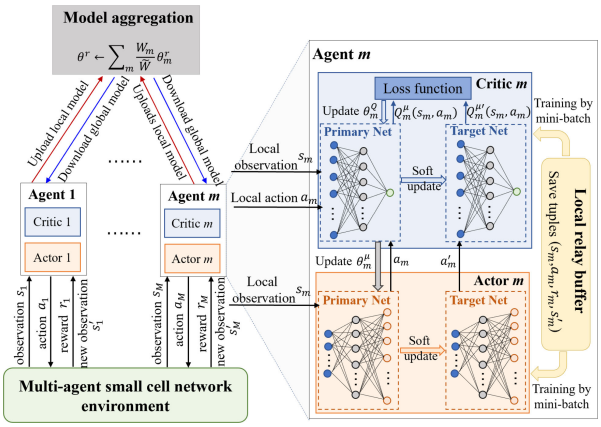

Fig. 3. The federated DRL scheme

inter-cell interference, each cell can individually decide its computation offloading and resource allocation policy if the interference from the other cells is acquired. Consequently, given the received interference on all channels, the global optimization problem P0 can be decomposed into M sub-problems, each corresponding to a small cell. Take cell m for instance, given $I_k^{(m)}[t]$ for $k \in \mathcal{K}$ in (1), the local optimization sub-problem can be formulated as

$$\text{P1:} \min_{\boldsymbol{\alpha}_m[t],\mathbf{b}_m[t],\mathbf{p}_m[t],\mathbf{f}_m[t]} \sum_n \left(LocE_n^{(m)}[t] + Off\,E_n^{(m)}[t] \right)$$

$$s.t.$$

$$\text{C1 - C7 in (11) for given } m \tag{22}$$

where $\boldsymbol{\alpha}_m[t] = \{\alpha_n^{(m)}[t]\}$, $\mathbf{b}_m[t] = \{b_{n,k}^{(m)}[t]\}$, $\mathbf{p}_m[t] = \{p_n^{(m)}[t]\}$, and $\mathbf{f}_m[t] = \{f_n^{(m)}[t]\}$ for $n \in \mathcal{N}_m, k \in \mathcal{K}$. Note that the sub-problem P1 is also an MINLP, with the goal of minimizing the energy consumption of the cell rather than the total energy consumption of all cells. In addition, all of the constraints are only related to the cell itself.

From the system perspective, when the global optimization problem is decomposed into multiple sub-problems, each SBS agent can independently learn a policy based on its own observation and interaction with the environment, while treating the other agents as part of the environment. However, the lack of training data may pose a significant challenge to the training of an accurate DRL model at each agent in independent learning. To deal with this challenge with low overhead, a distributed federated learning [28] can be exploited to enhance the training performance of the individual local DRL models without centralizing the training data [29].

As shown in Fig. 3, each SBS runs a local DDPG model, where the actor network takes as input its local observation of the environment and outputs the selected action, and the critic network takes as input the local observation and the selected action and outputs the corresponding Q-value. According to the local optimization problem (22), each SBS agent aims at minimizing the energy consumption of its own cell while satisfying the latency requirements of the tasks within its coverage. Therefore, the reward $R_m^{Fl}[t]$ for SBS agent m at time t can be defined as

$$R_m^{Fl}[t] = \sum_{n \in \mathcal{N}_m} \left(-LocE_n^{(m)}[t] - Off E_n^{(m)}[t] + V_n^{(m)}[t] \right) \tag{23}$$

where $V_n^{(m)}[t]$ is given by (13). Different from the reward $R_m^{Co}[t]$ in (12), the reward $R_m^{Fl}[t]$ is designed as the local utility of each cell.

For SBS agent m, we denote the actor and critic networks as $\boldsymbol{\mu}_m$ and Q_m parameterized by $\theta_m^{\boldsymbol{\mu}}$ and θ_m^Q, respectively. Moreover, SBS agent m maintains an experience replay buffer to store its local transition tuples $(s_m[t], a_m[t], R_m^{Fl}[t], s_m[t+1])$. With the experience replay strategy, the critic network is updated by minimizing the loss function $L_m^{Fl}(\theta_m^Q)$, which is defined as

$$L_m^{Fl}(\theta_m^Q) = \frac{1}{W_m} \sum_{j=1}^{W_m} \left[y_m^j - Q_m^{\boldsymbol{\mu}} \left(s_m^j, a_m^j \right) \right]^2 \tag{24}$$

where W_m is the size of mini-batches of SBS agent m. $y_m^j = R_m^{Fl} + \gamma Q_m^{\boldsymbol{\mu}'}(s'^j_m, a'^j_m)\,|_{a'^j_m = \boldsymbol{\mu}'_m(s'^j_m)}$ is the target value generated by the target critic network. Note that the action-value function $Q_m^{\boldsymbol{\mu}}(s_m, a_m)$ takes as input the observation and action of itself. Since the impact of other SBS agents' policies on the achievable local reward is characterized by the received interference $\mathbf{I}_m$ in the local observation of the environment, the critic network can evaluate the quality of the selected action based on its local information.

On the other hand, the actor network of SBS agent m can be updated with the policy gradient scheme as follows.

$$\nabla_{\theta_m^{\boldsymbol{\mu}}} J_m^{Fl}(\theta_m^{\boldsymbol{\mu}}) = \frac{1}{W_m} \sum_{j=1}^{W_m} \nabla_{\theta_m^{\boldsymbol{\mu}}} \boldsymbol{\mu}(s_m^j) \nabla_{a_m} Q_m^{\boldsymbol{\mu}}(s_m^j, a_m^j)|_{a_m^j = \boldsymbol{\mu}(s_m^j)} \tag{25}$$

Subsequently, the weights θ_m^Q and $\theta_m^{\boldsymbol{\mu}}$ can be updated with (17) and (19), respectively. In addition, the weights of the corresponding target critic and actor networks can be updated with (20).

The proposed federated DRL based computation offloading and resource allocation algorithm is summarized in Algorithm 2. In the proposed algorithm, the federated learning process consists of multiple coordination rounds. In each round, each SBS agent independently trains its local model based on its local data through multiple training episodes, without any knowledge of the observations and actions of other SBS agents. Consequently, there is no information exchange among SBS agents, and hence the communication overhead of the system is significantly reduced.

After training for multiple episodes on each round, each SBS agent uploads the weights of its local model to the coordinator (e.g., the macro base station (MBS)) via a dedicated backhaul control channel to perform model aggregation. Specifically, the mini-batch based stochastic gradient descent is adopted for federated averaging, wherein the weights of the global model are given by

$$\theta^r \leftarrow \sum_m \frac{W_m}{\tilde{W}} \theta_m^r \tag{26}$$

Algorithm 2: Federated DRL Based Computation Offloading and Resource Allocation Algorithm.

Initialize actor networks $\boldsymbol{\mu}$ and critic networks $\mathbf{Q}$ with weights $\boldsymbol{\theta^{\mu}}$ and $\boldsymbol{\theta}^{Q}$.
Initialize target actor networks $\boldsymbol{\mu'}$ and target critic networks $\mathbf{Q}'$ with weights $\boldsymbol{\theta^{\mu'}}$ and $\boldsymbol{\theta}^{Q'}$.
Initialize the experience replay buffer $\mathcal{D}_m$ for SBS agent $m \in \mathcal{M}$;
for each round $r = 1, 2, \ldots$ **do**
  for each episode $epi = 1, 2, \ldots$ **do**
    Initialize the local observation $s_m[0]$ for $m \in \mathcal{M}$;
    for each time $t = 1, 2, \ldots$ **do**
      All SBS agents select actions simultaneously $a_m[t] = \boldsymbol{\mu}_m(s_m[t]), m \in \mathcal{M}$, and execute the selected actions;
      for each SBS agent **do**
        Measure the received interference $I_k^{(m)}[t]$ on each channel $k \in \mathcal{K}$, and compute the immediate reward $R_m^{Fl}[t]$ with (23), and observe the next state $s_m[t+1]$;
        Save $(s_m[t], a_m[t], R_m^{Fl}[t], s_m[t+1])$ in $\mathcal{D}_m$;
        Sample a random mini-batch from $\mathcal{D}_m$;
        Update the critic network with (17) by minimizing the loss function in (24);
        Update the actor network with (19) and (25);
        Update the target networks with (20);
      end for
    end for
  end for
  All SBS agents upload the weights $\theta_m^{\boldsymbol{\mu}}$ and θ_m^{Q} of their local models to the coordinator;
  Perform model aggregation with (26), and distribute the averaged global weights back to all SBS agents;
end for

where θ^r and θ_m^r are the weights of the global model and the local model at SBS agent m on round r, respectively. $\tilde{W} = \sum_m W_m$ is the sum batch size for all SBS agents. Then, the coordinator distributes the averaged global model back to all SBS agents to update the their local models accordingly. Alternatively, the model aggregation can be performed at each SBS agent locally in implementation. In particular, the SBS agents exchange their model parameters with each other via an inter-cell dedicated control channel. Each SBS agent performs model aggregation with (26) based on the collected model parameters from the other agents. Since each SBS agent acquires the model parameters of all SBS agents and performs the same federated averaging operation, different SBS agents can obtain the same aggregated model as that performed by the MBS globally. Further, the neighbouring small cells may experience similar environment observations, the SBS agents of the neighbouring small cells can be clustered into a cooperating group to perform model aggregation, reducing the communication overhead.

Since each SBS agent also runs a local DDPG model, the time complexity of the proposed federated DRL algorithm can also be calculated using (21). Although the structure of the actor network is the same as that in the proposed multi-agent collaborative DRL algorithm, the input size of the critic network $n_{C,0}$ is significantly reduced since only the local state is considered instead of the observations and actions of all SBS agents. Therefore, the complexity of the proposed federated DRL algorithm is significantly reduced compared to the proposed multi-agent DRL algorithm.

V. Numerical Results

In this section, numerical results are presented to validate the proposed multi-agent DRL and federated DRL based computation offloading and resource allocation algorithms for small cell networks. To verify the performance of our proposed algorithms, we introduce the following three benchmark algorithms:

- *Independent learning (IL) [25] based algorithm*: The SBS agents are independent learners without any cooperation, where each SBS agent learns a policy based on its own observation and interaction with the environment, and there is no model sharing and information exchange among the SBS agents.
- *Full offloading scheme:* All computation tasks are offloaded to the corresponding serving SBSs for remote computing. In each cell, each UE transmits its task to the SBS on a randomly assigned channel with its maximum transmit power. The SBS equally allocates its computation resource to the UEs.
- *Local computing scheme:* Each UE executes its computation task locally.

A. Simulation Setup

We considered an area of 1.5 km $\times$ 1 km with 6 randomly deployed SBSs, where the cell radius is 250 m, and 10 UEs are randomly placed in each small cell. We modeled the wireless channel as the block Rayleigh fading channel followed by the Okumura-Hata path loss model, i.e., $PL(d) = 137.74 + 35.22 \log(d)$ in dB, where d is the distance between the transmitter and the receiver in kilometers. There are 16 orthogonal channels in the system, and the channel bandwidth is 10 MHz. The maximum transmit power of each UE is 23 dBm, and the AWGN spectrum density is -174 dBm/Hz. The size of the computation task is randomly distributed between 200 KB and 300 KB, and the CPU cycles required for per bit data is randomly distributed between 500 and 1000. We assumed the maximum latency of each UE is equal to ninety percent of the local latency. The computation capacities of each UE and each SBS are 1 GHz and 50 GHz, respectively. The energy consumption coefficient of each UE is 1 J/Gcycle, and that of each SBS is 10 mJ/Gcycle.

The adopted DDPG for each SBS agent in the simulation is a three-layer fully connected neural network with one hidden layer, each consisting of 64 neurons. The rectified linear unit (ReLU) is utilized as the activation function and Adam optimizer is used to update network weights with a learning rate of 0.001. The size of experience replay buffer is set to 10 000, and the size of minibatch is set to 32. The discount factor is set to 0.9, and the soft updating rate of target networks is set to 0.01.

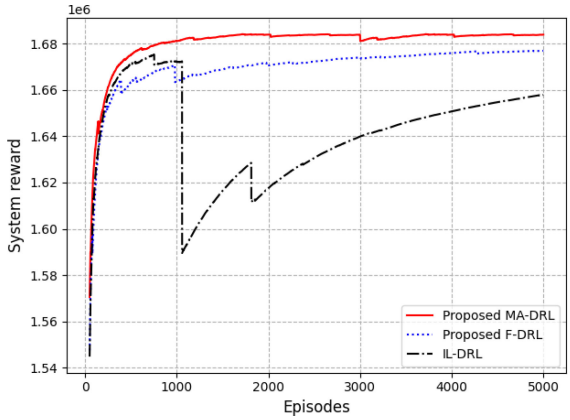

Fig. 4. Convergence performance of different algorithms

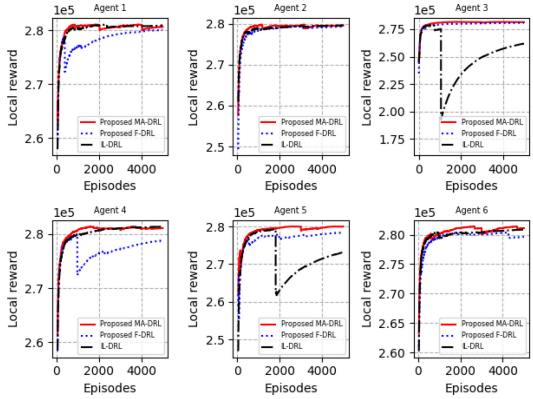

Fig. 5. Local reward of each agent in different algorithms

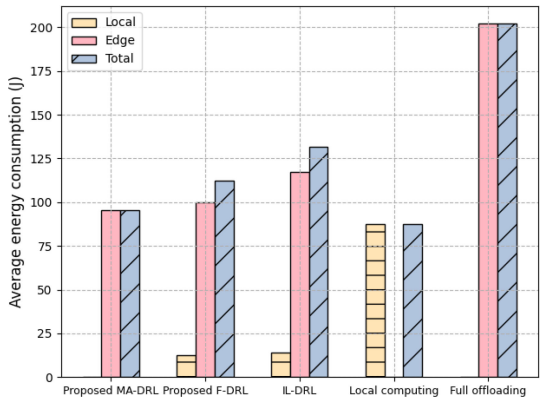

Fig. 6. Average energy consumption for different algorithms

B. Performance Analysis

Fig. 4 compares the convergence performance of different algorithms in terms of system reward, which is the sum of local rewards obtained by all SBS agents in the system. We observe that the proposed multi-agent DRL algorithm (denoted as "MA-DRL") achieves the highest system reward and its convergence is the most stable compared to the other two algorithms. This stems from the fact that the proposed MA-DRL algorithm leverages extra information about the observations and actions of other SBS agents to facilitate the training process, thus improving the stability of the training process as well as the system reward. It reveals that the cooperation between the SBS agents in the training process can effectively overcome the inherent nonstationarity of the multi-agent environment thus contributing in improving the system performance. At the same time, we also observe that the proposed federated DRL based algorithm (denoted as "F-DRL") outperforms the independent learning based algorithm (denoted as "IL-DRL"). In the proposed F-DRL and the IL-DRL algorithms, each SBS agent trains its model based on its own observation and action, without exchanging its local data with the other agents. However, the federated learning by sharing the parameters of local models among the different SBS agents can improve the training performance of the local model, thus achieving higher system reward.

To gain insight into the impact of different learning methods on the performance of the individual local models, we show the local reward obtained by each SBS agent for different algorithms in Fig. 5. It can be seen that the SBS agents exhibit different learning behaviours due to different local observations and policies. In particular, the difference between the local rewards obtained by different SBS agents in the proposed MA-DRL algorithm is smaller than that in the other two algorithms. This can be explained as follows. The system performance-oriented reward function is applied to guide the SBS agents to improve their policies, leading to the balanced performance among different SBS agents. On the other hand, Fig. 5 also shows that even for the same SBS agent, its learning curves are distinct in different algorithms. Generally, with the proposed MA-DRL algorithm, the SBS agents can achieve higher local reward with better stability compared to the other two algorithms.

Fig. 6 compares the energy consumption in different algorithms. In addition to the total energy consumption, the energy consumptions of local computing and edge computing are shown in the figure. First, the total energy consumptions in MA-DRL and F-DRL are lower than that in the independent learning based algorithm and full offloading scheme, but higher than that in the local computing scheme. Compared with the local computing scheme, the proposed MA-DRL and F-DRL algorithms intelligently offload the tasks to the corresponding serving SBS, which introduces an extra energy consumption for communications, resulting in higher total energy consumption. Nevertheless, Fig. 7 shows that the proposed MA-DRL and F-DRL algorithms dramatically improve the demand satisfaction rate at a cost of the reasonable increase in energy consumption. Compared with the full offloading scheme, the proposed MA-DRL and F-DRL algorithms considerably reduce energy consumption by jointly optimizing computation offloading, channel allocation, power control, and computation resource allocation. Compared to the IL-DRL algorithm, our proposed MA-DRL algorithm ensures that the SBS agents learn better policies by making use of the global information about the observations, actions, and rewards of all SBS agents to train their models. Moreover, our proposed F-DRL algorithm enhances the performance of the individual local models by exploiting model sharing among SBS agents, whereas the models may be affected by the limited local training data in the IL-DRL algorithm. Thus, the cooperation among

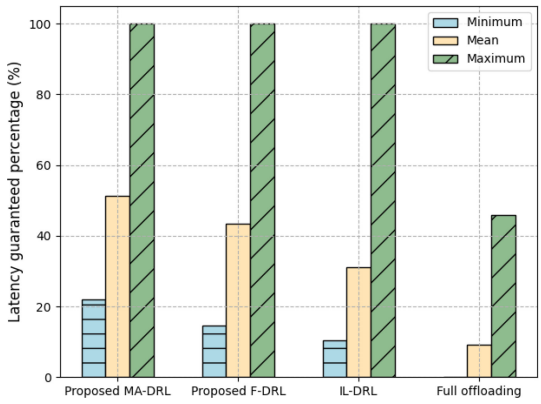

Fig. 7. Latency guaranteed percentage for different algorithms

SBS agents can improve the policies of agents in a multi-agent environment, thus improving the overall energy consumption performance of the system. Second, the average energy consumptions of local computing in the proposed MA-DRL algorithm is zero, whereas local computing consumes energy in the proposed F-DRL and the IL-DRL algorithms. This implies that SBS agents have learnt different strategies utilizing these three intelligent algorithms, rendering different performance, which corroborates the results show in Fig. 5. Third, the average total energy consumptions in the proposed F-DRL algorithm is slightly higher than that in the proposed MA-DRL algorithm. The reason is that learning based on local observation and action may degrade the training performance of the model, although federated learning is employed.

Fig. 7 shows the statistics of the latency guaranteed percentage in different algorithms. The latency guaranteed percentage is the ratio of UEs that meet their task latency requirements. Since the maximum latency of each UE is equal to ninety percent of the local latency, the latency guaranteed percentage of the local computing scheme is always zero, so it is omitted in the figure. Note that the original optimization problem P0 in (11) may be infeasible due to the limited communication and computation resources of the system, such that it is impossible to satisfy the latency requirements of all UEs. In such cases, the latency guaranteed percentage is an important metric to measure the ability of guaranteeing task requirements.

Fig. 7 shows that the proposed MA-DRL and F-DRL algorithms significantly outperforms the benchmark algorithms. In particular, the proposed MA-DRL algorithm achieves the highest average latency guaranteed percentage. Compared with the local computing scheme and the full offloading scheme, the proposed MA-DRL and F-DRL algorithms can intelligently perform the joint optimization of computation offloading and resource allocation, thus improving the latency guaranteed percentage. We would also like to emphasize that even when the system is in a restricted situation, the proposed MA-DRL and F-DRL algorithms can still guarantee the latency requirements of some UEs, while the two benchmark algorithms fail. Compared to the IL-DRL algorithm, the proposed MA-DRL and F-DRL algorithms facilitate the SBS agents to learn better strategies by utilizing the cooperation among SBS agents. From Fig. 7, we also note that the performance of the proposed F-DRL algorithm is close to that of the proposed MA-DRL algorithm. The average latency guaranteed percentage in the proposed F-DRL algorithm is about 85% of that in the proposed MA-DRL algorithm. Compared to proposed MA-DRL algorithm, the proposed F-DRL algorithm only requires the SBS agents to share their model parameters rather than the local training data, greatly reducing the signalling overhead.

VI. Conclusion

In this paper, we investigated the joint optimization problem of computation offloading and interference coordination for small cell networks. To reap the performance gains brought by cooperation among agents, we proposed the multi-agent DRL based computation offloading and resource allocation algorithm to minimize total energy consumption while ensuring the latency requirements of the tasks. In addition, we proposed the federated DRL scheme to reduce the computing complexity and signalling overhead due to the training process. Numerical results clearly indicated that our proposed schemes can considerably improve the system performance in terms of energy consumption and latency guaranteed percentage compared with the local computing scheme, the full offloading scheme, and the independent learning based algorithm.

References

[1] Y. Teng, M. Liu, F. R. Yu, V. C. M. Leung, M. Song, and Y. Zhang, "Resource allocation for ultra-dense networks: A survey, some research issues and challenges," *IEEE Commun. Surv. Tut.*, vol. 21, no. 3, pp. 2134–2168, Jul.-Sep. 2019.

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

[3] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.

[4] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, "A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2762–2773, Dec. 2018.

[5] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense iot networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.

[6] A. Samanta and Z. Chang, "Adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3864–3872, Apr. 2019.

[7] B. Yang, X. Cao, X. Li, C. Yuen, and L. Qian, "Lessons learned from accident of autonomous vehicle testing: An edge learning-aided offloading framework," *IEEE Wireless Commun. Lett.*, vol. 9, no. 8, pp. 1182–1186, Aug. 2020.

[8] W. Sun, P. Wang, N. Xu, G. Wang, and Y. Zhang, "Dynamic digital twin and distributed incentives for resource allocation in aerial-assisted internet of vehicles," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2021.3058213.

[9] Z. Zhao *et al.*, "A novel framework of three-hierarchical offloading optimization for MEC in industrial IoT networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5424–5434, Aug. 2020.

[10] H. Khelifi *et al.*, "Bringing deep learning at the edge of information-centric Internet of Things," *IEEE Commun. Lett.*, vol. 23, no. 1, pp. 52–55, Jan. 2019.

[11] B. Yang *et al.*, "Edge intelligence for autonomous driving in 6 G wireless system: Design challenges and solutions," 2020, *arXiv:2012.06992v1*.

[12] W. Sun, S. Lei, L. Wang, Z. Liu, and Y. Zhang, "Adaptive federated learning and digital twin for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5605–5614, Aug. 2021.

[13] B. Yang, X. Cao, C. Yuen, and L. Qian, "Offloading optimization in edge computing for deep learning enabled target tracking by Internet-of-UAVs," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9878–9893, Jun. 2021.
[14] Y. Dai, D. Xu, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning and permissioned blockchain for content caching in vehicular edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4312–4324, Apr. 2020.
[15] K. Zhang, J. Cao, H. Liu, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for social-aware edge computing and caching in urban informatics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5467–5477, Aug. 2020.
[16] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7635–7647, Oct. 2019.
[17] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, 2019.
[18] Y. Xie, Z. Xu, J. Xu, S. Gong, and Y. Wang, "Backscatter-aided hybrid data offloading for mobile edge computing via deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn. Intell. Commun.*, 2019, pp. 525–537.
[19] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5 G beyond," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12175–12186, Oct. 2020.
[20] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2019.2902661.
[21] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for stochastic computation offloading in digital twin networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4968–4977, Jul. 2021.
[22] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2681–2690.
[23] Y. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2239–2250, Oct. 2019.
[24] Z. Li and C. Guo, "Multi-agent deep reinforcement learning based spectrum allocation for D2D underlay communications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1828–1840, Feb. 2020.
[25] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 729–743, Feb. 2020.
[26] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. 11th Int. Conf. Mach. Learn.*, vol. 157, 1994, pp. 157–163.
[27] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multiagent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, Long Beach, CA, 2017, pp. 1–12.
[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[29] W. Sun, N. Xu, L. Wang, H. Zhang, and Y. Zhang, "Dynamic digital twin and federated learning with incentives for air-ground networks," *IEEE Trans. Netw. Sci. Eng.*, to be published, doi: 10.1109/TNSE.2020.3048137.

Xiaoyan Huang received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2012. She is currently an Associate Professor with the School of Information & Communication Engineering, University of Electronic Science and Technology of China. Her research interests include edge intelligence, wireless communications and networking, and Internet of Things.

Supeng Leng (Member, IEEE) received the Ph.D. degree from Nanyang Technological University, Singapore, in 2005. He is currently a Professor with the School of Information & Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include resource, spectrum, energy, routing and networking in wireless sensor networks, broadband wireless access networks, smart grid, and vehicular networks.

Sabita Maharjan (Senior Member, IEEE) received the Ph.D. degree in networks and distributed systems from the University of Oslo and Simula Research Laboratory, Norway, in 2013. She is currently a Senior Research Scientist with the Simula Metropolitan Center for Digital Engineering, Norway, and an Associate Professor (adjunct position) with the University of Oslo. Her current research interests include vehicular networks and 5G, network security and resilience, smart grid communications, Internet of Things, machine-to-machine communication, software defined wireless networking, and advanced vehicle safety.

Yan Zhang (Fellow, IEEE) received the B.S. degree from Beihang University, Beijing, China, the M.S. degree from the Nanjing University of Post and Telecommunications, Nanjing, China, and the Ph.D. degree from the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. He is currently a Full Professor with the Department of Informatics, University of Oslo, Oslo, Norway. His research interests include next-generation wireless networks leading to 6G and green and secure cyber-physical systems, such as smart grid and transport.

He is a Fellow of IET, an Elected Member of Academia Europaea (MAE), the Royal Norwegian Society of Sciences and Letters (DKNVS), the Norwegian Academy of Technological Sciences (NTVA). Since 2018, he was the recipient of the global Highly Cited Researcher Award (Web of Science top 1% most cited worldwide). He is the Symposium or Track Chair of a number of conferences, including the IEEE ICC 2021, IEEE SmartGridComm 2021, and IEEE GLOBECOM 2017. He is the Chair of IEEE Communications Society Technical Committee on Green Communications and Computing (TCGCC). He is the Editor (or Area Editor, Senior Editor, an Associate Editor) for several IEEE Transactions/magazine, including the IEEE NETWORK MAGAZINE, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE COMMUNICATIONS SURVEY & TUTORIALS, IEEE INTERNET OF THINGS JOURNAL, IEEE SYSTEMS JOURNAL, IEEE VEHICULAR TECHNOLOGY MAGAZINE, and IEEE Blockchain Technical Briefs. He is a Distinguished Lecturer of IEEE Communications Society and a Distinguished Speaker of IEEE Vehicular Technology Society. From 2016 to 2020, he was also an Distinguished Lecturer of IEEE Vehicular Technology Society.