Multiagent DDPG-Based Joint Task Partitioning and Power Control in Fog Computing Networks

Zhipeng Cheng, Minghui Min, Minghui Liwang, *Member, IEEE*, Lianfen Huang, and Zhibin Gao, *Member, IEEE*

***Abstract*—Fog computing is an energy-efficient and cost-effective paradigm to help alleviate the pressure of resource-constrained mobile devices (MDs) running computation-intensive applications. In this article, we investigate the joint task partitioning and power control problem in a fog computing network with multiple MDs and fog devices (FDs), where each MD has to complete a periodic computation task under the constraints of delay and energy consumption. Each task can be partitioned into multiple subtasks and offloaded to the FDs according to the task partition strategy and transmission power strategy to reduce task execution delay and energy consumption. To this end, we present a multiagent deep deterministic policy gradient (MADDPG)-based task offloading algorithm for MDs to maximize the long-term system utility including the execution delay and energy consumption. Each MD inputs the local information, e.g., the task requirements, the available communication, and computation resources of the FDs, the computation resources, and the battery level of the MD into a distributed actor network to generate a task offloading policy, while a centralized critic network is used to update the weights of the actor networks to improve offloading performance. Numerical simulation results demonstrate the effectiveness of the proposed scheme in improving the system utility, reducing the average execution delay as well as the average energy consumption.**

***Index Terms*—Fog computing, multiagent deep deterministic policy gradient (MADDPG), task offloading, task partitioning, power control.**

I. Introduction

ALONG with the springing-up emergence of computation-intensive and delay-sensitive applications, such as online 3-D gaming, video surveillance, and augmented/virtual realities [1]–[3], overburdened mobile devices (MDs) struggle to meet the stringent requirements of the computational tasks. Offloading tasks to a centralized distant cloud server with sufficient computation resources is a feasible way, which, however, will lead to heavy service delays caused by the long-distance and bottleneck effect of fronthaul and backhaul networks. Over the recent years, fog computing has become a popular paradigm, by utilizing the communication and computation resources of different fog devices (FDs), such as neighboring base stations (BSs), access points (APs), mobile-edge computing (MEC) servers, and end devices (e.g., vehicles, laptops) [4]. However, due to the computation resources constraints and heterogeneity of the FDs [7], the conventional binary offloading [5], which offloads the whole task to an FD, may cause a high task failure rate. To this end, partial offloading is a more resource-efficient and energy-efficient way [6]. In partial offloading, an MD can partition the computation task into several subtasks and offload the subtasks to different FDs for parallel execution [7].

Generally, the task offloading problems can be roughly divided into two categories, according to the users involved: 1) single-user task offloading problem [8], [9] and 2) multiuser task offloading problem [7], [10], [11], where the latter is more complicated due to the competition or cooperation among the users. Thus, the optimal offloading policy of a multiuser task offloading problem can not simply be superimposed by the optimal policy of the single-user model. Typically, the multiuser task offloading problem can be modeled as a multiplayer offloading game [12]. Nevertheless, in a multiplayer offloading game, the computation complexity may grow exponentially with the number of users and FDs, and it is difficult to obtain the global optimal policy without the global information. Thus, it is challenging to derive the theoretical Nash equilibrium[13], [14]. Moreover, most existing works formulate the problem based on a one-shot optimization and ignore the ever-changing network environments, such as the channel gains, the available resources of all FDs, which is hard to predict and model. The problem needs to be resolved when the environment changes. Therefore, the frequent recalculation of the optimal policies has become the pivotal disincentive of the existing methods.

Recently, deep reinforcement learning (DRL)-based task offloading has attracted significant attention from academia and industry [15]–[19], [21], [26], [27]. In particular, single-agent DRL (SADRL)-based task offloading problem

Manuscript received February 5, 2021; revised April 13, 2021 and May 27, 2021; accepted June 15, 2021. Date of publication June 22, 2021; date of current version December 23, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61871339, Grant 61971365, Grant 61731012, and Grant 61971368; in part by the Natural Science Foundation of Fujian Province of China under Grant 2019J01003; and in part by the National Natural Science Foundation of China under Grant U20A20162. This article was presented in part at the GLOBECOM 2020—2020 IEEE Global Communications Conference, Taipei, Taiwan, December 2020. *(Corresponding author: Zhibin Gao.)*

Zhipeng Cheng, Lianfen Huang, and Zhibin Gao are with the School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: chengzp_x@163.com; lfhuang@xmu.edu.cn; gaozhibin@xmu.edu.cn).

Minghui Min is with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China (e-mail: minmh@cumt.edu.cn).

Minghui Liwang is with the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON N6A 3K7, Canada (e-mail: mliwang@uwo.ca).

Digital Object Identifier 10.1109/JIOT.2021.3091508

was studied in [15]–[19], while [21], [26], [27] studied the multiagent DRL (MADRL)-based task offloading problem. Compared with SADRL, MADRL suffers from an unstable learning environment. In MADRL, each agent should not only consider the dynamics of the environment itself but also take the actions of other agents into account. Therefore, from the perspective of a single agent, other agents are part of the environment. When all agents are learning to change their policies, the environment is unstable and training convergence can not be guaranteed. Moreover, due to the lack of global information sharing and central control in a multiagent system, each agent makes a decision only according to its local information without knowing the actions and the states of other agents, which necessitates an effective learning framework.

To address the challenges of MADRL, a novel policy-based multiagent deep deterministic policy gradient (MADDPG) algorithm is proposed in [22]. MADDPG works in a centralized training and decentralized execution paradigm and can be used for multiagent continuous decision making problem, which exactly fits the task partitioning and power control problem investigated in this article.

In this article, we study the joint task partitioning and power control-based multiuser partial offloading problem in a fog computing network consists of multiple MDs and FDs. Each MD partitions the computation task into several subtasks according to the available communication and computation resources of the FDs, the computation resources and the battery level of the MD, and the task requirements, and decides the transmission power to transmit the subtasks to the FDs accordingly. Then the subtasks are collaboratively executed by the MD and FDs to reduce the execution delay as well as the execution energy consumption while respecting the delay and energy constraints of the task.

We first model this problem as a multiagent stochastic offloading game. Then an MADDPG-based task offloading algorithm is proposed to make the offloading decision for each MD, which includes the task partition strategy and transmission power strategy. Specifically, the main contributions of this article are summarized as follows.

1) We investigate the joint task partitioning and power control problem in a fog computing network with multiple MDs and FDs. The joint task partitioning and power control problem is regarded as a continuous decision making problem to maximize the long-term system utility while taking the task execution delay and energy consumption of each MD into consideration.
2) Considering the dynamics of the environment and the high-dimensional action space of the task partitioning and power control, we transform the problem as a multiuser stochastic offloading game, and resolve the problem based on an MADDPG model. In MADDPG, each MD is modeled as an agent and only shares the local information and the observation of the environment to a centralized critic network, while makes decisions locally with a distributed actor network.
3) Extensive simulations are conducted to demonstrate the system performance under different computation task sizes, number of FDs, and number of MDs. We compare the system performance with three baseline schemes, simulation results reveal that our proposed MADDPG-based task offloading algorithm can greatly improve the system utility, reduce the average execution delay as well as the average energy consumption.

The remainder of this article is organized as follows. We review the related work in Section II and demonstrate the system model and formulate the multiuser task offloading problem as a long-term system utility maximization problem in Section III. The MADDPG-based task offloading algorithm is presented in Section IV and extensive simulations are conducted in Section V. Finally, we conclude this article in Section VI.

Notation: For ease of presentation, in this article, the lowercase boldface letters are used to denote vectors, e.g., $\mathbf{x}$, while $\nabla\omega J$ stands for the partial derivative of function J with respect to ω.

II. Related Work

Compared with binary offloading, task partitioning-based partial offloading can fully exploit the advantages of parallel execution by utilizing the computation resources of multiple devices at the same time [23]. Some early efforts have denoted to the single-user partial offloading scenario in [6] and [8]–[10]. In [6], a cooperation framework of cloud computing and MEC is considered, where the user's application can be partitioned into several modules and can be either offloaded to the remote server (i.e., cloud or MEC) or executed locally. Al-Habob *et al.* [8] and Liu and Zhang [9] studied the task offloading problem in a MEC system, where the task can be split into subtasks and offloaded to different servers. They formulated the problem as a weighted-sum optimization problem of delay and offloading failure probability, and the problem is solved by a heuristic algorithm. In [10], the joint problem of partial offloading and power allocation is formulated in a single-user MEC system to minimize the weighted sum of execution delay and energy consumption.

The multiuser scenario is investigated in [6], [7], [24], and [28]. In [6], the partial offloading problem is studied based on the cooperation framework of MEC and cloud computing. They solved the single-user offloading problem first and extended it to the multiuser offloading problem with an iterative heuristic algorithm. Sheng *et al.* [7] studied the joint problem of task partitioning, radio resource allocation, and computation resource allocation in a multiuser single-MEC system to minimize the energy consumption of the MD. A dynamic matching algorithm is proposed to adapt to the heterogeneity of the network. Cao *et al.* [24] investigated the task partitioning problem in a multiuser multiMEC system, and jointly optimized the task dispatching and scheduling strategy to minimize the execution time, and they proved in theory that the largest task should be dispatched first while the shortest task should be executed first on the edge server to improve offloading performance. Liu *et al.* [28] studied the task partitioning problem in a heterogeneous fog computing network which consists of multiple task nodes, and helper nodes, each task can be partitioned into several subtasks and offloaded

to different help nodes for parallel execution to minimize the execution delay. They formulated the problem as a generalized Nash equilibrium problem and derived the Nash equilibrium.

However, the joint problem of task partitioning and resource allocation in a multiuser multiserver system can be very complicated while taking the heterogeneity and dynamics of network resources into consideration. Besides, users can not make the optimal offloading decision due to the asymmetrical network information [18], which also promotes the research of the RL-based task offloading problem. A DRL-based mobile offloading scheme is proposed in [16], where an MD can choose an edge server and decide the offloading rate, the transmission power to improve its utility as well as against jamming and interference. The joint task offloading and resource allocation problem is considered in a multifog network in [19], each fog node cooperates to maximize the local reward only according to local observations. Thus, they formulated this problem as a partially observable stochastic game, and they applied a deep recurrent Q-network to approximate the optimal offloading value functions. The nonorthogonal multiple access (NOMA) assisted multitask computation offloading in multiaccess MEC is investigated in [20]. Considering the time-varying channel power gains, a DRL-based online algorithm is proposed to learn the optimal NOMA-transmission duration to minimize the total energy consumption of local device for task execution under the delay constraint. A noncooperative offloading game is considered in a system that consists of multiple MDs and edge devices in [21]. They aimed to design an optimal offloading policy to minimize the task drop rate as well as the execution delay, without the known prior knowledge of the task arrival models and channel models. An MADDPG-based algorithm is proposed to the task offloading and multichannel access problem in [26], where each user makes an offloading decision to decide the task is executed locally or offloaded to the cloud server by selecting a wireless channel for communication. A novel imitation learning-based task offloading algorithm is proposed in [27], where each MD can mimic the task offloading policy of the experts without knowing the global information.

III. System Model

A. Network Model and Task Model

We consider a heterogeneous fog computing network composed of M randomly distributed FDs[1] with diverse communication and computation resources. There are N MDs[2] with limited computation resources and battery level, and each MD needs to complete a computation-intensive and delay-sensitive task periodically. The system time is slotted with a constant slot duration, and the time slot index can be denoted by t, where $t \in \{1, 2, \ldots, T\}$. Furthermore, according to the general assumption in many works [9], [28], the quasi-static scenario is also considered in this article. The important notations that will be used in the following are listed in Table I.

TABLE I
Summary of Symbols and Notations

Notation	Definition
M/N	Number of FDs/MDs in the fog network
$\mathcal{M}/\mathcal{N}$	Set of FDs/MDs
$E_i(t)$	Battery level of MD i
$B_i(t)$	Input data size of MD i
$C_i(t)$	Required total CPU cycles of MD i
β	Number of CPU cycles to process one bit
$T_i(t)$	Maximum delay tolerance of MD i
$\mathbf{y}_i(t)$	Task partition strategy of MD i
$y_{i,j}(t)$	Task partition ratio from MD i to FD j
$\mathbf{w}(t)$	Available communication resources of all FDs
$w_j(t)$	Available communication resources of FD j
$r_{i,j}(t)$	Data rate from MD i to FD j
$\lambda_{i,j}(t)$	Communication / Computation resources allocation ratio allocated to MD i from FD j
$\mathbf{p}_i(t)$	Transmission power strategy of MD i
$p_{i,j}(t)$	Transmission power from MD i to FD j
$f_i^l(t)$	Computation resources of MD i
$\mathbf{f}^s(t)$	Available computation resources of all FDs
$f_j^s(t)$	Computation resources of FD j
$\tau_i(t)$	Total execution delay of MD i
$e_i(t)$	Total energy consumption of MD i

Denote the set of FDs and MDs by $\mathcal{M} = \{1, 2, \ldots, M\}$, and $\mathcal{N} = \{1, 2, \ldots, N\}$, respectively. At time slot t, we assume each MD has a splittable task that needs to be executed with respect to the delay and the energy consumption constraints. We assume the battery level of the MD i at the time slot t can be denoted as $E_i(t) \in [\underline{E}_i, \overline{E}_i]$, where $\underline{E}_i$ and $\overline{E}_i$ are the minimum and the maximum battery level, respectively. To avoid running out of battery, the battery level can be renewed at different time slots by applying an energy harvesting technology, such as the charging model proposed in [17]. However, please note that in this article we only care about the changes in battery level and do not specifically depend on a given charging model.

The task of MD i at time slot t can be described by a tuple $(B_i(t), C_i(t), T_i(t))$, where $B_i(t)$ is the input data size, in terms of bits. $C_i(t) = \beta B_i(t)$ is the CPU cycles required for processing this task, where β is the computation intensity that indicates the required CPU cycles per bit. $T_i(t)$ is the maximum delay tolerance. In this article, we assume that the system time slot length is greater than or equal to the maximum delay tolerance of all tasks. The advantage of this assumption is to avoid the cross-slot optimization due to the diversity of task delay requirements. Furthermore, in this article, all MDs and FDs are required to have an ideal synchronization state. When the slot length is large, the impacts of synchronization error caused by hardware limitation can be reduced. Cross-slot optimization or subslot optimization to study the impacts of synchronization error can be further analyzed as future work.

As shown in Fig. 1, to reduce the execution delay as well as the energy consumption, an MD can partition the computation task into several subtasks. The subtasks can be offloaded to the FD in parallel and are collaboratively executed by the MD and the FDs. We assume the FD allocates the communication

[1]The FD mentioned in this article is the abstraction of all devices that can provide computing services for other devices.

[2]The MD metioned in this article is the abstraction of devices that need to offload tasks.

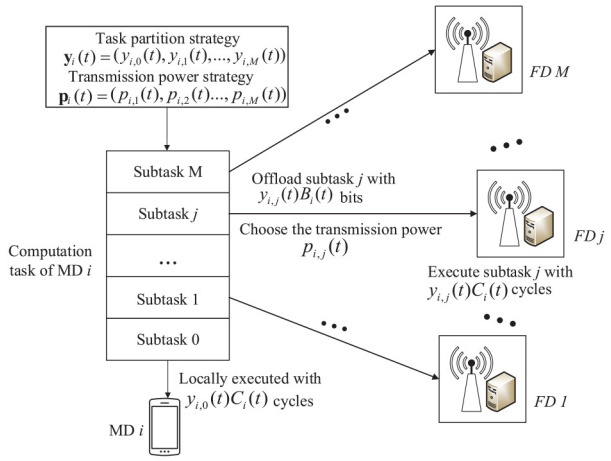


Fig. 1. MD i partitions the computation task with task partition strategy $\mathbf{y}_i(t)$ and offloads the subtasks to M FDs with transmission power strategy $\mathbf{p}_i(t)$.

and computation resources in proportion to MD's workload, which is also applied in [25] and [28].

At time slot t, MD i partitions the computation task into $M+1$ nonoverlapping subtasks. Let $\mathbf{y}_i(t) = (y_{i,0}(t), y_{i,1}(t), \ldots, y_{i,M}(t))$ be the task partition strategy of MD i, where $0 \le y_{i,j}(t) \le 1$, $\sum_{j\in\mathcal{M}} y_{i,j}(t) + y_{i,0}(t) = 1\ \forall i \in \mathcal{N}$. $y_{i,0}(t)$ is the partition ratio for local execution, and $y_{i,j}(t)$ is the partition ratio of the subtask that will be offloaded to FD j from MD i. $\mathbf{w}(t) = (w_1(t), w_2(t), \ldots, w_M(t))$ is the available communication resources (i.e., bandwidth) of all FDs at time slot t. Assume $g_{i,j}$ is the channel gain from MD i to FD j, and remains constant within one time slot. Thus, the uplink data rate from MD i to FD j at time slot t can be calculated as

$$r_{i,j}(t) = \lambda_{i,j}(t) w_j(t) \log\left(1 + \frac{p_{i,j}(t) g_{i,j}}{N_0(t)}\right) \tag{1}$$

$$\lambda_{i,j}(t) = \frac{y_{i,j}(t) B_i(t)}{\sum_{i=1}^{N} y_{i,j}(t) B_i(t)} \quad \forall j \in \mathcal{M} \tag{2}$$

where $\lambda_{i,j}(t)$ is the proportion of communication resources allocated by FD j to MD i, $\sum_{i=1}^{N} y_{i,j}(t) B_i(t)$ is the total input data size (workload) offloaded to FD j. $0 < p_{i,j}(t) \le p_{\max}$ is the transmission power of MD i to FD j, $p_{\max}$ is the maximum transmission power of the MD. $N_0(t)$ is the noise power.

B. Execution Model

In this section, we will discuss the delay and energy consumption of local execution and fog execution.

1) Local Execution: Let $f_i^l(t)$ denote the processing speed of MD i, in terms of cycles/s. Thus, the local execution delay of MD i is

$$\tau_i^l(t) = \frac{y_{i,0}(t) C_i(t)}{f_i^l(t)}. \tag{3}$$

The local execution delay should not exceed the delay tolerance $T_i(t)$, i.e., $y_{i,0}(t)$ should meet $y_{i,0}(t) \le ([T_i(t) f_i^l(t)]/[C_i(t)])$.

As for the local execution energy consumption, we consider the widely adopted model of energy consumption per computing cycle as $\mathcal{E} = \kappa c^2$, where c is the CPU frequency and κ is the energy coefficient related to the chip architecture [29]. Thus, the energy consumption for local execution of user i can be calculated as

$$e_i^l(t) = \kappa \left(f_i^l(t)\right)^2 y_{i,0}(t) C_i(t). \tag{4}$$

Again, the local execution energy consumption should not exceed the battery level constraint, as a result, $y_{i,0}(t)$ should satisfy $y_{i,0}(t) \le ([E_i(t)]/[\kappa (f_i^l(t))^2 C_i(t)])$. Finally, we have $y_{i,0}(t) \le \min\{([T_i(t) f_i^l(t)]/[C_i(t)]), ([E_i(t)]/[\kappa (f_i^l(t))^2 C_i(t)])\}$.

2) Fog Execution: The delay of the offloading process comprises three parts: 1) the uplink transmission delay to transmit the input data to the FD; 2) the execution delay at the FD; and 3) the downlink transmission delay to send the output results from the FD back to the MD. Generally, the downlink has a much higher data rate than that of the uplink, and the output is in small data size. Thus, the downlink transmission delay can be ignored, which is also assumed in [21] and [28].

When the FD receives the subtasks from different MDs, it will allocate the computation resource to execute the subtask on behalf of the MD. Let $\mathbf{f}^s(t) = (f_1^s(t), f_2^s(t), \ldots, f_M^s(t))$ denote the available computation resources of all FDs at time slot t, expressed in terms of the number of CPU cycles per second.

The transmission delay and energy consumption for MD i to offload its subtask to FD j can be calculated as

$$\tau_{i,j}^{\text{stran}}(t) = \frac{y_{i,j}(t) B_i(t)}{r_{i,j}(t)} \tag{5}$$

$$e_{i,j}^{\text{stran}}(t) = p_{i,j}(t) \tau_{i,j}^{\text{stran}}(t). \tag{6}$$

Generally, the FDs are directly powered by the power grid, and yet the MD is energy-limited, so we do not consider the energy consumption caused by the task execution at the FDs.

Similarly, the computation resource allocated to MD i by FD j is $\lambda_{i,j}(t) f_j^s(t)$. Then the subtask execution delay of MD i at FD j is

$$\tau_{i,j}^{\text{sexe}}(t) = \frac{y_{i,j}(t) \mathrm{C}_i(t)}{\lambda_{i,j}(t) f_j^s(t)}. \tag{7}$$

Finally, the total delay experienced by the subtask of MD i when it is offloaded to FD j is

$$\tau_{i,j}^{so}(t) = \tau_{i,j}^{\text{stran}}(t) + \tau_{i,j}^{\text{sexe}}(t). \tag{8}$$

When all the subtasks executed at the MD and at the FDs are finished, the task can be regarded as completed. Thus, the total delay of MD i is defined as

$$\tau_i(t) = \max\left(\tau_i^l(t), \max_{j\in\mathcal{M}}\left(\tau_{i,j}^{so}(t)\right)\right) \quad \forall i \in \mathcal{N}. \tag{9}$$

The total energy consumption of MD i is

$$e_i(t) = e_i^l(t) + \sum_{j\in\mathcal{M}} e_{i,j}^{\text{stran}}(t). \tag{10}$$

C. Problem Formulation

In this article, we aim to optimize the task offloading policy of each MD to reduce the execution delay and energy consumption, where the MD can decide the task partition strategy

and the transmission power strategy to offload the subtasks to the FDs. To evaluate the task offloading performance of MD, we define the utility of MD i at time slot t as [16]

$$u_i(t) = \alpha_b B_i(t) - \alpha_d \tau_i(t) - \alpha_e e_i(t) \tag{11}$$

where α_b indicates the unit reward for successfully processing one bit of the task, $B_i(t)$ is the total executed bits (i.e., the input data size of MD i as listed in Table I) of the MD and FDs. It is worth noting that the first item is 0 if the task fails, which is also in line with the requirements of the realistic scenario. α_d is unit cost of the execution delay while α_e is unit cost of the energy consumption.

Then, we define the long-term utility of MD i as the weighted summation of instantaneous utility over a finite period T

$$\Phi_i(t) = \sum_{t=1}^{t=T} \gamma^{(t-1)} u_i(t) \tag{12}$$

where $\gamma \in [0, 1]$ is the discount factor to determine the future reward.

When the MDs offload the subtasks to the same FD, they have to compete for the communication and computation resources. Thus, the execution delay and energy consumption of MD i depend on the joint task partition strategies $(\mathbf{y}_i(t), \mathbf{y}_{-i}(t))$ and the transmission power strategy $\mathbf{p}_i(t)$, where $\mathbf{y}_{-i}(t)$ is the task partition strategies of all MDs except MD i.

Finally, we can formulate the long-term total system utility maximization problem as

$$\max_{(\mathbf{y}_i(t), \mathbf{y}_{-i}(t)), \mathbf{p}_i(t)} : \sum_{i \in \mathcal{N}} \Phi_i(t) \tag{13}$$

$$\text{s.t.} \quad y_{i,j}(t) \in [0, 1], i \in \mathcal{N}, j \in \mathcal{M} \tag{13a}$$

$$\sum_{j \in \mathcal{M}} y_{i,j}(t) + y_{i,0}(t) = 1, i \in \mathcal{N} \tag{13b}$$

$$\sum_{i \in \mathcal{N}} \lambda_{i,j}(t) = 1, j \in \mathcal{M} \tag{13c}$$

$$\sum_{j \in \mathcal{M}} p_{i,j}(t) \le p_i^{\max}, i \in \mathcal{N} \tag{13d}$$

$$\tau_i(t) \le T_i(t), i \in \mathcal{N} \tag{13e}$$

$$e_i(t) \le E_i(t), i \in \mathcal{N}. \tag{13f}$$

Constraints (13a) and (13b) indicate that the subtasks are collaboratively executed by the MD and FDs. Constraint (13c) ensures the communication and computation resources allocated to the MDs do not exceed the total available resources of the FD. Constraint (13d) guarantees the total transmission power cannot exceed the maximum transmission power of the MD. Constraints (13e) and (13f) enforce that the constraints of the delay tolerance and energy consumption are not violated.

Problem (13) is difficult to solve for three reasons. First, (13) is a nonlinear programming problem, which is generally NP-hard; second, the optimal task partition strategy of MD i not only depends on its own strategy but also depends on the strategies of the other MDs; third, the action space of the joint strategies increases exponentially with the number of the MDs and the FDs. Thus, this problem can no more be tackled via traditional methods. Therefore, in the next section, a MADRL-based method, in particular, an MADDPG-based method, is proposed to solve this problem.

IV. Multiagent DDPG-Based Task Offloading

In this section, we will show how to utilize the MADDPG to learn the task partition strategy and the transmission power strategy of each MD. In detail, we first introduce the preliminaries of MADDPG, then the state, action, and reward of the MADDPG are defined for the MDs. Note that, for the sake of simplicity, some symbols use superscript t to replace (t) in the following.

A. Preliminaries of MADDPG

In the traditional SADRL framework, the agent learns to improve its strategy by interacting with the environment. Thereby the optimal strategy can be obtained from the environment. However, in a MADRL system, from the perspective of an individual agent, the environment is complex and dynamic, which brings great difficulties to the learning process. To this end, as an extension of DDPG, MADDPG is proposed to guarantee the convergence of learning performance.

DDPG is an RL framework that can be applied for continuous action control, which combines the deep Q-network (DQN) and the actor–critic algorithm. DDPG comprises two main online networks: 1) the actor network and 2) the critic network. The actor network is used to represent the deterministic policy gradient, and the critic network is used to approximate the Q-value function. Similar to DQN, both the actor network and the critic network have two subnetworks with the same network structure: 1) the online network and 2) the target network. Given the current state s^t, the action a^t, and the deterministic policy μ, the actor network and the target actor network are denoted as $\mu(s^t|\omega)$ and $\mu'(s^t|\omega')$, while the critic network and the target critic network are denoted as $Q(s^t, a^t|\theta)$ and $Q'(s^t, a^t|\theta')$, respectively. The ω, ω', θ, and θ' are the corresponding parameters of these four neural networks.

Generally, MADDPG works in a centralized training and decentralized execution manner. Thus, the global information is used in the training process, as long as this information is not used at the testing time [22]. For ease of understanding, Fig. 2 illustrates the overall framework of MADDPG with N agents. Suppose the continuous policy set of N agents is $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_N\}$, and $\boldsymbol{\omega} = \{\omega_1, \ldots, \omega_N\}$ is the parameter set of the corresponding policy. Agent i aims to maximize its expected reward with the objective function, e.g., $J(\omega_i) = \mathbb{E}[\Phi_i]$. Each agent can make independent decision according to the local observation, i.e., $a_i^h = \mu_{\omega_i}(o_i^h)$. The continuous policy μ_i of agent i is optimized by the gradient of the objective function with respect to ω_i as

$$\nabla_{\omega_i} \mathcal{J}(\mu_i) \approx \frac{1}{H} \sum_h \nabla_{\omega_i} \mu_i\left(o_i^h\right) \nabla_{a_i} Q_i^{\theta_i}\left(\mathbf{s}^h, a_1^h, \ldots, a_N^h\right)\Big|_{a_i^h = \mu_i(o_i^h)} \tag{14}$$

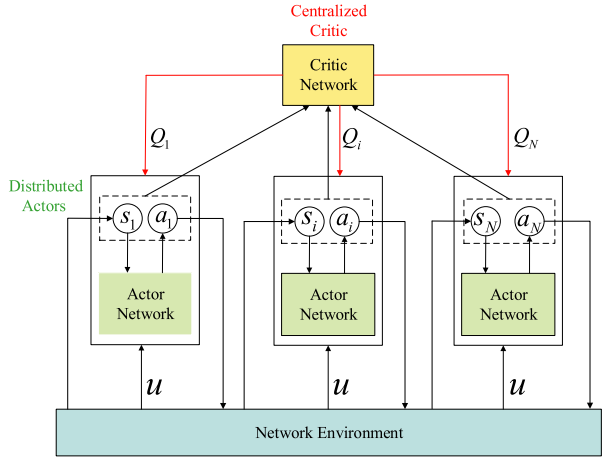


Fig. 2. Overall framework of the MADDPG with N agents.

where $Q_i^{\theta_i}(\mathbf{s}^h, a_1^h, \ldots, a_N^h)$ is a centralized action-value function, where $\mathbf{s}^h = (o_1^h, \ldots, o_N^h)$ contains the observations (states) of all N agents, $\mathbf{a}^h = (a_1^h, \ldots, a_N^h)$, u^h, and $\mathbf{s}^{h+1}$ are the actions, global reward, and new states of all agents, respectively. By taking advantage of the experience reply to break up the temporal correlations of the training samples, the tuples $(\mathbf{s}^h, \mathbf{a}^h, u^h, \mathbf{s}^{h+1})$ are stored in the centralized replay buffer. H is the size of the stochastically sampled mini-batch. In this way, the centralized critic network can obtain the joint state and the joint action of all agents.

Then, the centralized action-value function of agent i is updated according to the loss function

$$\mathcal{L}(\theta_i) = \frac{1}{H}\sum_h \left(y^h - Q_i^{\theta_i}\left(\mathbf{s}^h, a_1^h, \ldots, a_N^h\right)\right)^2 \tag{15}$$

$$y^h = u^h + \gamma Q_i^{\boldsymbol{\theta}_i'}\left(\mathbf{s}^{h+1}, a_1^{h+1}, \ldots, a_N^{h+1}\right)\Big|_{a_i^{h+1}=\boldsymbol{\mu}_i'\left(o_i^{h+1}\right)}. \tag{16}$$

Finally, each agent i updates the target networks as follows:

$$\begin{aligned}\theta_i' &\leftarrow \varsigma\theta_i + (1-\varsigma)\theta_i' \\ \omega_i' &\leftarrow \varsigma\omega_i + (1-\varsigma)\omega_i'.\end{aligned} \tag{17}$$

B. Multiagent Stochastic Game Formulation

In this section, we first formulate the multiagent stochastic game by defining the model as follows.

1) *Agents:* All N MDs.
2) *Action:* At time slot t, each agent decides its task partition strategy and corresponding transmission power strategy according to the current state. Then, the action of agent i is denoted as, $\mathbf{a}_i^t = (\mathbf{y}_i(t), \mathbf{p}_i(t))$.The joint action of N agents can be denoted as $\mathbf{a}^t = \{\mathbf{a}_1(t), \mathbf{a}_2(t), \ldots, \mathbf{a}_N(t)\}$.
3) *State*: At time slot t, agent i observes its local information from the network environment, i.e., the available communication resources $\mathbf{w}(t)$, and computation resources $\mathbf{f}^s(t)$ of the FDs, the local computation resource $f_i^l(t)$, battery level $E_i(t)$, and task requirement $\mathbf{s}_i^{tr}(t) = \{B_i(t), C_i(t), T_i(t)\}$ of the MD. Thus, the state of agent i can be defined as $\mathbf{o}_i^t = \{\mathbf{w}(t), \mathbf{f}^s(t), f_i^l(t), E_i(t), \mathbf{s}_i^{tr}(t)\}$. Then, the joint state of all agents at time slot t is $\mathbf{s}^t = (\mathbf{o}_1(t), \mathbf{o}_2(t), \ldots, \mathbf{o}_N(t))$.
4) *Reward:* In MADDPG, each agent cooperates to maximize the system utility. Thus, we can define the reward of all agents by sharing the same utility function defined in (13), denoted as u^t.

Algorithm 1 MADDPG-Based Task Offloading Algorithm for N MDs

Initialize the learning rate of the actor network and the critic network, the discount factor γ, the maximum learning episode EP, the maximum training steps T per episode.
Initialize the replay buffer D, the weights of actor and critic networks, random process Ψ.
Initialize the network layout with N MDs and M FDs.
for episode $= 1 : EP$ **do**
  Each MD receives the initial state $\mathbf{o}_i^t$ according to the task requirements and network environment, and the global state $\mathbf{s}^t$.
  for each step $t = 1 : T$ **do**
    For each agent i, takes action $a_i^t = \mu_{\omega_i}(\mathbf{o}_i^t) + \Psi^t$
    Execute the actions $\mathbf{a}^t = \{a_1^t, \ldots, a_N^t\}$ in the environment, receive the global reward u^t, next state $\mathbf{s}^{t+1}$.
    Store $(\mathbf{s}^t, \mathbf{a}^t, u^t, \mathbf{s}^{t+1})$ in replay buffer D.
    Set $\mathbf{s}^t \leftarrow \mathbf{s}^{t+1}$.
    for agent $i = 1 : N$ **do**
      Randomly sample H samples from D to make up a mini-batch.
      Set y^h according to (16).
      Update the critic network and the actor network via (15) and (14), respectively.
    end for
    Update the target networks via (17).
  end for
end for

Based on the defined state, action, and reward, we give the MADDPG-based task offloading algorithm for N MDs in Algorithm 1. More specifically, Fig. 3 illustrates the framework of our MADDPG-based task offloading algorithm. As shown in Fig. 3 and Algorithm 1, agent i observes the environment, and receives its initial state $\mathbf{o}_i^t$, then obtains the task partition strategy and the transmission power strategy by inputting the state into the actor network. To improve the exploration efficiency, a noise Ψ^t is sampled from an Ornstein-Uhlenbeck (OU) process is added to $\mu_{\omega_i}(\mathbf{o}_i^t)$ to generate temporally correlated exploration in the learning process [30]. After that, all agents execute their corresponding task offloading actions, and observe the reward u^t, joint next state $\mathbf{s}^{t+1}$. Then the agents store $(\mathbf{s}^t, \mathbf{a}^t, u^t, \mathbf{s}^{t+1})$ in a centralized replay buffer. Repeat until the replay buffer is full, then randomly sample H samples from the replay buffer to make up a mini-batch to train the actor networks, and the centralized critic network, as described in (14)–(17).

According to the analysis in [31]–[33], the time complexity of DRL offline training and online execution is dominated

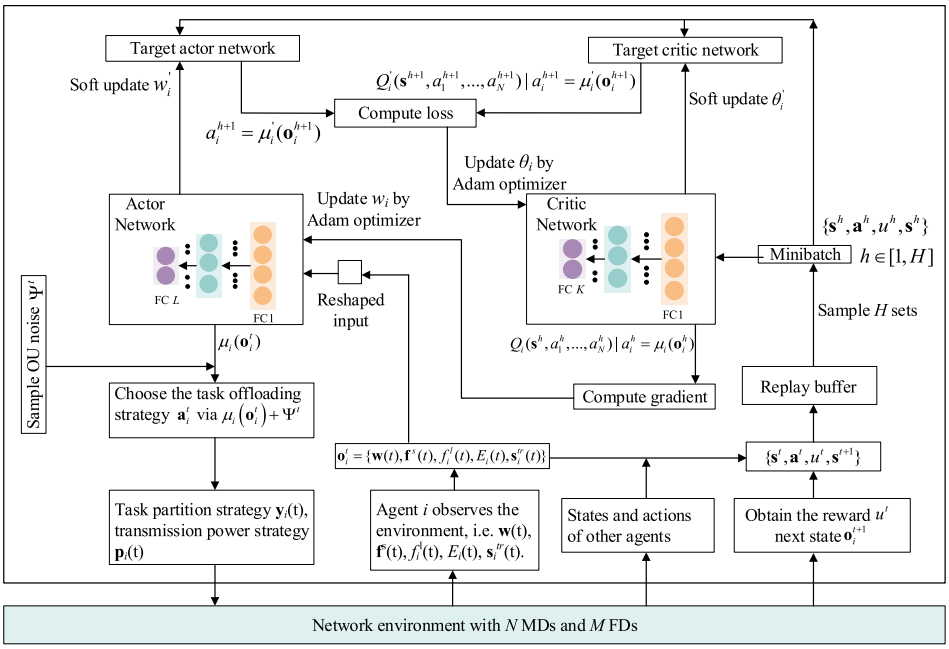


Fig. 3. Illustration of the MADDPG-based task offloading algorithm: from the perspective of agent i.

by the number of matrix multiplications. In particular, if the actor network and the critic network of each agent contain L and K fully connected (FC) layers, respectively, the single round iteration time complexity of training the DDPG model is $\mathcal{O}(\sum_{l=0}^{L-1} n_l n_{l+1} + \sum_{k=0}^{K-1} n_k n_{k+1})$, where n_l represents the neuron number of layer l. The time complexity of online execution (i.e., execution of actor network) is $\mathcal{O}(\sum_{l=0}^{L-1} n_l n_{l+1})$.

Although N agents need to be trained during the training process, the time complexity can be reduced by using multithreaded programming to make full use of the multicore CPU. Nevertheless, the centralized training process of the MADDPG is energy-consuming and resource-hungry, so, instead of using an MD, we can conduct the training on an FD [34]. When the model is trained on an FD, there will be some additional transmission overhead, including the transmission of training data from the MDs to the FD and the distribution of the trained model from the FD to the MDs. The MADDPG model considered in this article is relatively lightweight, and the size of the training data and the model are within tens of kB, so the additional transmission overhead can be ignored compared to the cost of the task offloading process. When the training data is in large volume, some state-of-the-art learning mechanisms can be used to further reduce the transmission overhead for future work, such as multiagent federated learning or transfer learning [35].

V. Numerical Simulation

In this section, we evaluate the performance of our proposed MADDPG-based task offloading algorithm through comprehensive simulations and compare the performance with different baseline schemes.

A. Simulation Settings

We consider a simulation scenario with three MDs and three FDs by default. Each MD generates a computation task at the beginning of each time slot, We assume that the input task size ranges from 500 to 1100 at 100-kb intervals, and the computation intensity $\beta = 1000$ cycles/bit by default. For simplicity, we assume all MDs have constant local computation resource as $f_i^l = 1$ GHz, and $\kappa = 10^{-28}$ [17]. The battery level of all MDs can be decided by the energy model proposed in [17], the minimum battery level is 50 mJ and the maximum battery level is 100 mJ. The maximum delay tolerance of all MDs is set to 500 ms. The maximum transmission power of all MDs is set to 20 dBm, and we assume the normalized channel gain from MD i to FD j follows a uniform distribution with $g_{i,j} \sim U[0.3, 0.8]$, where the reference channel gain is -20 dB. The noise power density is -174 dBm/Hz. The unit reward of processing one bit $\alpha_b = 2 * 10^{-5}$.

The FDs have diverse available communication and computation resources, and we model the communication resource of FD i as a Markov chain with $\Pr(w_i(t+1) = z \mid w_i(t) = z') = P_{zz'}$ $\forall z, z' \in \{2, 3, 4, 5\}$ MHz. Similarly, the available computation resource can be modeled a Markov chain with $\Pr(f_i(t+1) = z \mid f_i(t) = z') = P_{zz'}$ $\forall z, z' \in \{2, 3, 4, 5\}$ GHz.

Each agent in our proposed MADDPG algorithm contains a DDPG network architecture as shown in Table II. We use three FC networks in the actor network at most, with 256, 128, and 64 neurons, respectively. The ReLu activation function is used in the hidden layer. The output is a vector of size $1 * (2M + 1)$, while the first $1 + M$ items represent the task partition strategy, and the last M items stand for the transmission power strategy. The critic network also has 256, 128, and 64 neurons in FC 1-3, respectively. We conduct the simulation on Intel Core i7-7700 3.60-GHz CPU with 16-GB RAM. The important hyperparameters are listed in Table III.

TABLE II
NETWORK ARCHITECTURE OF THE MADDPG

Network	Layer	Size	Activation
Actor network	FC 1	256	
	FC 2	128	
	FC 3	64	ReLu
Critic network	FC 1	256	
	FC 2	128	
	FC 3	64	ReLu

TABLE III
HYPERPARAMETERS OF THE MADDPG

Hyperparameter	Value
Episode EP	200
Steps T	1000
γ	0.95
Replay buffer size	4000
Optimizer	Adam
Python version	3.6
Learning framework	Tensorflow 1.15.0

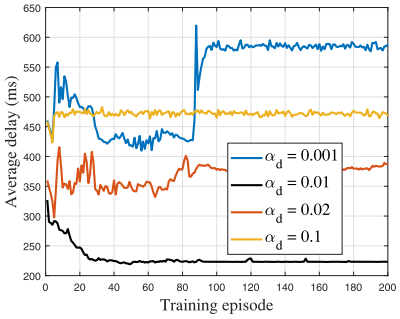
To evaluate the performance of our proposed MADDPG-based task offloading algorithm, we use three different benchmark schemes, as follows.

1) *Full Local (FL):* All MDs decide to locally execute their tasks.
2) *DDPG:* Each MD has an independent DDPG model, and there is no centralized critic network and local information sharing, each MD makes an independent decision.
3) *DDRLO [16]:* We modify the DRL-based mobile offloading scheme proposed in [16]. In the DDRLO scheme, each MD chooses only one FD from multiple FDs, and decides the task partition ratio, the transmission power accordingly based on a DDPG model. The essential difference between DDPG and DDRLO is that DDPG allows MDs to select multiple FDs. Thus, the DDRLO scheme can be regarded as a typical partial offloading baseline scheme.

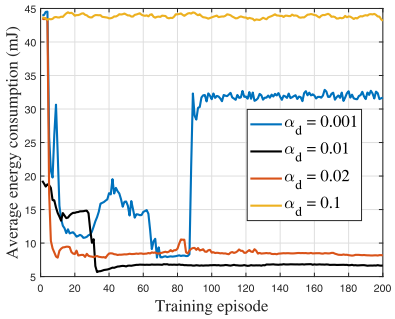
B. Simulation Results

1) Impacts of Weights: The utility function defined in (11) indicates that the utility depends on the tradeoff between the delay and energy consumption. Thus, we evaluate the influence of different delay and energy consumption weights (i.e., α_d and α_e) of the utility function in Fig. 4. We first set the weight of energy consumption to 1 and set the weight of delay ranging from 0.001 to 0.1.

Fig. 4 curves the average delay and average energy consumption of the training performance. First, it can be seen that all curves finally converge to different values, and it shows that our simulation has robust convergence performance for different weights. Specifically, when α_d equals 0.001, it has the highest average delay with 590 ms and the average energy consumption is 32 mJ. Moreover, when α_d equals 0.1, it has the highest average energy consumption with 44 mJ, and the average delay is 470 ms. This indicates that it does not work well when α_d is too large or too small. So we evaluate the system performance with $\alpha_d = 0.01$ and $\alpha_d = 0.02$. We can see that it has the lowest average delay and average energy consumption when α_d equals to 0.01, and it converges in the 60th episode, which is also faster than other curves. Thus, we set $\alpha_e = 1$ and $\alpha_d = 0.01$ in our following simulations.



(a)



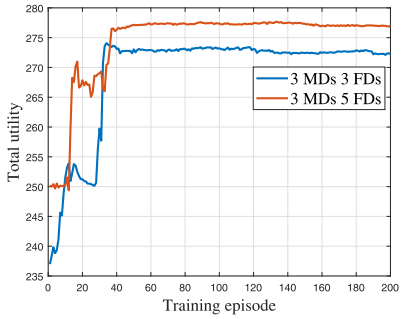
(b)

Fig. 4. System performance and convergence of training with different delay and energy consumption weights. (a) Average delay. (b) Average energy consumption.

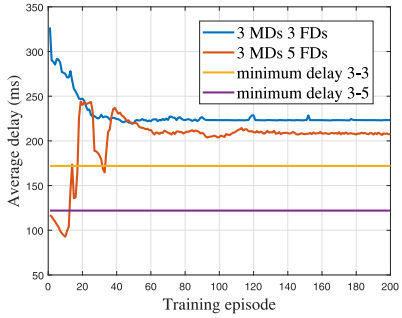
2) Convergence Performance: The system performance over the training episode when there are three MDs, three FDs, and five FDs is shown in Fig. 5. It can be seen that the two curves in the three subfigures converge almost simultaneously. However, we can see that in Fig. 5(a), the total utility of five FDs is slightly higher (about 2%) than that of three FDs. Accordingly, the average delay and average energy consumption of five FDs are lower than those of three FDs. It indicates that increasing the number of neighboring FDs may increase the total utility, reduce the average delay and average energy consumption, which will be further analyzed in Fig. 10.

Furthermore, the minimum achievable average delay can be obtained by setting the weight of the energy consumption to 0, which means that we only try to minimize the delay performance and ignore the energy consumption. As shown in Fig. 5(b), the minimum delay of three FDs is 23% lower than the original average delay, while the minimum delay of five FDs reduces the average delay by 41%. Similarly, we can obtain the minimum achievable average energy consumption by setting the weight of the delay to 0, which is shown in Fig. 5(c). Particularly, the average energy consumption is reduced by about 13% and 21%, respectively.

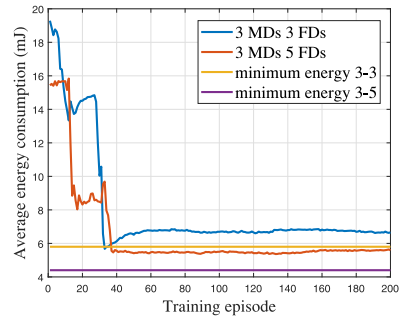
Fig. 6 illustrates the individual utility of three MDs when there are three FDs and five FDs. For simplicity, we denote three MDs, three FDs by 33, and 35 stands for three MDs and five FDs in the legend. It can be seen that the utility of



(a)



(b)



(c)

Fig. 5. System performance over the training episode when there are three MDs, with three FDs, and five FDs, respectively. (a) Total utility. (b) Average delay. (c) Average energy consumption.

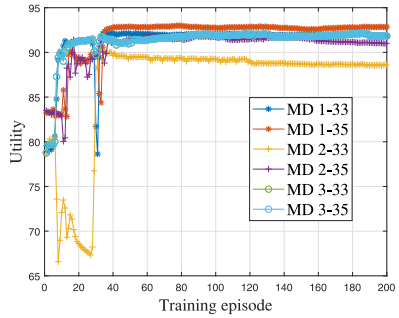


Fig. 6. Individual utility performance over the training episode when there are three MDs, with three FDs, and five FDs, respectively.

each MD eventually converges to a stable value. Moreover, each MD has a similar utility under the same number of FDs. It indicates that each MD can eventually reach an equilibrium state [36] under the framework of our proposed scheme, while also ensuring fairness among the MDs.

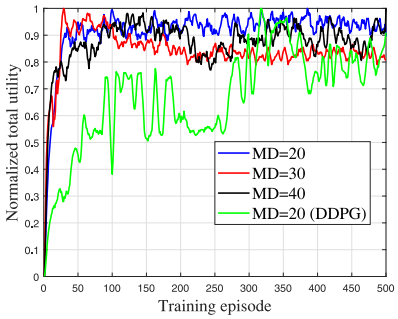
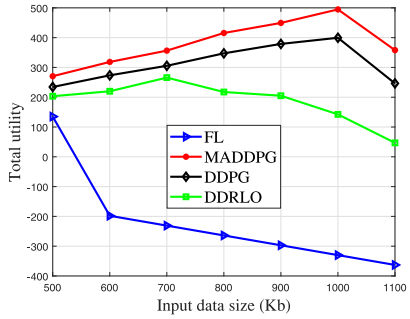


Fig. 7. Convergence performance of the MADDPG and the DDPG over the training episode in a medium-scale fog computing network, with 20 FDs and the number of MD ranges from 20 to 40.

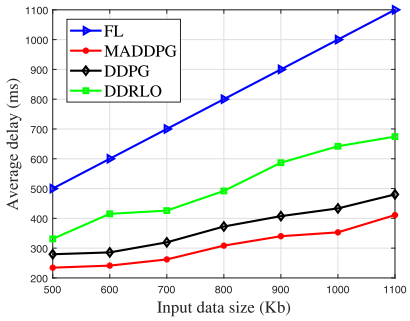
To further illustrate the adaptability of our proposed scheme in a medium-scale network, we further analyze the convergence performance in Fig. 7. First, it can be seen from Fig. 7 that our proposed MADDPG-based scheme still has better convergence performance when the number of MD ranges from 20 to 40. Besides, it can be observed that as the number of MD increases, the episode required for convergence is also longer. For example, the MADDPG converges after 200 episodes with 30 MDs. However, due to the lack of a collaboration mechanism in the DDPG, when the number of MDs is 20, the DDPG can no longer converge, which further proves the reliability of our proposed scheme. In this article, due to the limitations of the problem itself, such as the number of subtasks that can be partitioned and hardware limitations, large-scale or ultralarge-scale fog computing networks (with hundreds/thousands of MDs and FDs) are not considered.

However, due to inherent defects of the MADDPG algorithm, such as the centralized-decentralized mismatch issue caused by the centralized training and decentralized execution paradigm [37]. This problem will worsen as the number of agents increases. Therefore, our proposed scheme should be further optimized in future work before it can be applied to ultralarge-scale networks. To facilitate analysis and comparison with other algorithms, in the following simulations, we mainly perform simulations in small-scale networks.

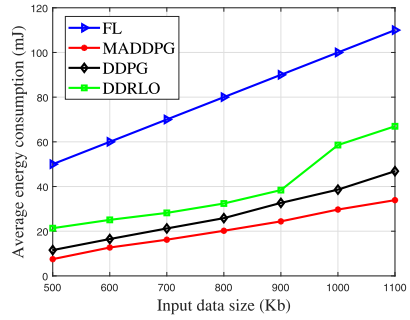
3) Impacts of Input Data Size: Fig. 8 compares the system performance averaged over the time slots with three MDs and three FDs for the 500–1100-kb tasks of different schemes. First, it is obvious to see that the FL has the lowest total utility, the highest average delay and average energy consumption. Specifically, when the input data size is larger than 500 kb, the FL only obtains negative utility. This is because both the average delay and average energy consumption exceed the delay and battery level constraints, and MDs fail to locally execute the tasks. It also can be seen that our MADDPG-based scheme outperforms the other three schemes with the highest total utility, the lowest average delay and average energy consumption. And it can be noticed that the DDPG has a better performance compared with DDRLO with higher utility, shorter average delay, and less average energy consumption. This indicates that better performance may be obtained by offloading subtasks to multiple FDs for collaborative execution. Furthermore, the total utility of the MADDPG and the



(a)

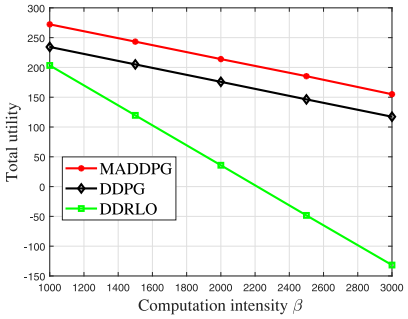


(b)

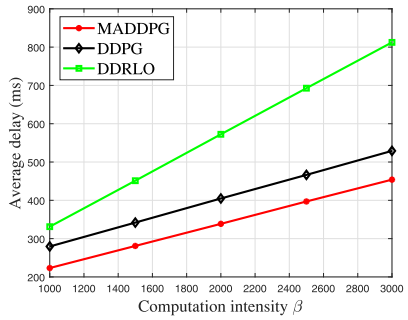


(c)

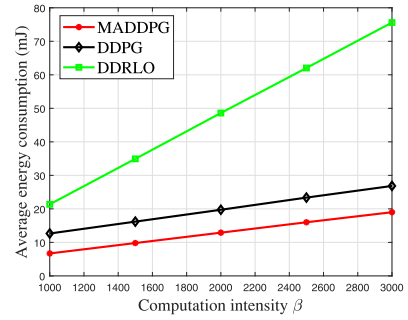
Fig. 8. Performance of different schemes versus the input data size of the task. (a) Total utility. (b) Average delay. (c) Average energy consumption.



(a)



(b)



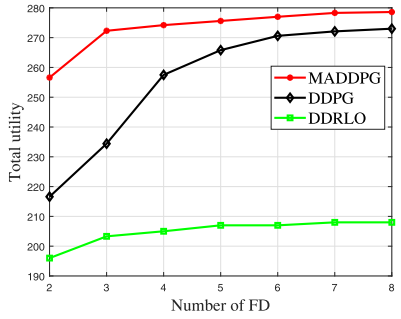
(c)

Fig. 9. Performance of different schemes versus the computation intensity β of the task. (a) Total utility. (b) Average delay. (c) Average energy consumption.

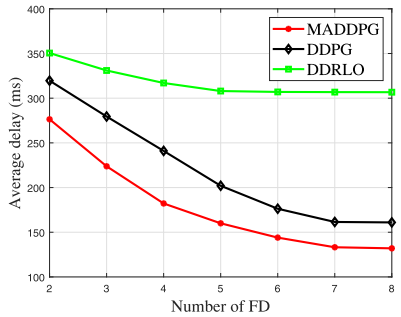
DDPG reach their peak at 1000 kb and begin to decline at 1100 kb, while the DDRLO peaks at 700 kb. The rationale behind this is that due to the limitation of the communication resources and computation resources, the upper bound of the system capacity is reached at 1100 kb. In general, Fig. 8 proves that the average delay and average energy consumption can be greatly reduced by offloading tasks to the FDs, while the system performance can be further improved by the collaboration mechanism of the MADDPG through centralized training and distributed execution.

4) Impacts of Computation Intensity: Fig. 9 compares the system performance averaged over the time slots with three MDs and three FDs for the 500-kb tasks with computation intensity β ranging from 1000 to 3000 of the different schemes. First, it can be seen that the total utility of the three schemes decreases with the increase of β, and the average delay, as well as the average energy consumption, increases with the increase of β. This is because the unit delay, as well as the unit energy consumption increases with an increase of β, while the unit reward for successfully processing one bit of the task is the same. However, it also can be seen that the MADDPG still outperforms the other two schemes. Specifically, compared with the DDPG and the DDRLO, the MADDPG increases the total utility by 32% and 218% when $\beta = 3000$, decreases the average delay by 14% and 44%, and decreases the average energy consumption by 29% and 75%, respectively.

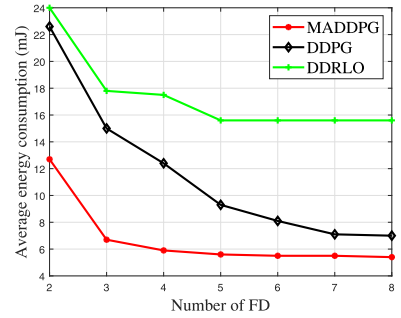
5) Impacts of FD Number: Fig. 10 compares the system performance averaged over the time slots with three MDs while the number of FD is ranging from 2 to 8. Again, we can see that the MADDPG still outperforms the other two schemes in the total utility, the average delay, and the average energy consumption. However, the common feature of the three curves in Fig. 10 is that their total utility increases initially with the increase of the FD number and then tends to be stable, while the average delay and average energy consumption begin to decrease until they become stable. In particular,



(a)

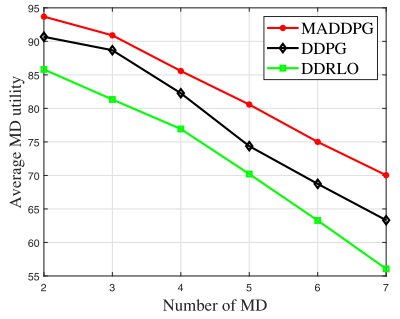


(b)

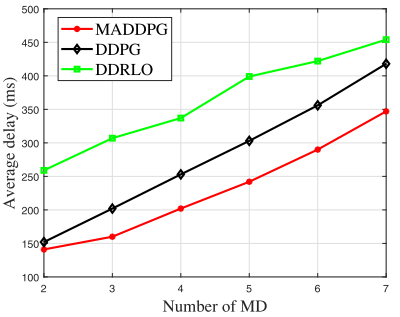


(c)

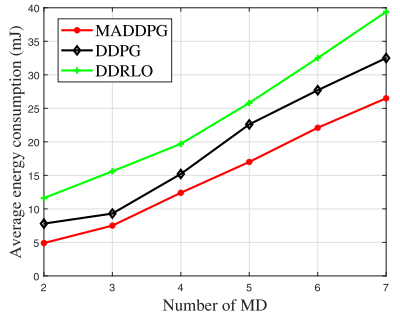
Fig. 10. Performance of different schemes versus the number of FD. (a) Total utility. (b) Average delay. (c) Average energy consumption.



(a)



(b)



(c)

Fig. 11. Performance of different schemes versus the number of MD with five FDs. (a) Average MD utility. (b) Average delay. (c) Average energy consumption.

the average delay of the MADDPG is reduced by 52% from 276.4 ms to 132 ms when the number of FD increases from 2 to 8 and begins to stabilize afterward. Meanwhile, the average energy consumption decreases by 57% from 12.7 mJ to 5.4 mJ. Similar trends can be observed in the other two schemes. It is worth noting that the DDRLO tends to be stable when the number of FD is 5 and then changes slightly afterward. This is because only one FD can be selected at a time in DDRLO, and the available resources of the other FDs can not be fully utilized, which is also consistent with the simulation results in [16]. It is also noteworthy that the performance gap between the MADDPG and the DDPG is getting smaller as the number of FD increases. For example, the average energy consumption of the MADDPG is 43.8% lower than that of the DDPG when the number of the FD is 2, but only 23% lower when the number of FD is 8. The reason behind this is that with the increase in the number of FD, more FDs with sufficient resources are likely to appear. Therefore, the DDPG is also more likely to learn a better strategy.

6) Impacts of MD Number: Fig. 11 illustrates the system performance averaged over the time slots with five FDs while the number of MD is ranging from 2 to 7. Instead of using the total utility, we evaluate the average MD utility in Fig. 11(a) for two reasons. First, theoretically, the total utility increases with the increase of the number of MD, as each MD can get a positive utility in the FL scheme, which can also be inferred from Fig. 7. Second, it makes more sense to care about the average performance with the number of MD increases, which can be used to measure the system capacity. It can be seen intuitively from Fig. 11 that the average MD utility decreases with the number of MD increases, while the average delay and average energy consumption increases. It is noteworthy that different from Fig. 10, the performance gap between the MADDPG and the DDPG increases with the number of MD increases. In more detail, the average MD utility of the MADDPG is only 9% higher than that of the DDPG when the number of MD is 2, while the MADDPG increases the

average MD utility by 25% when the number of MD is 7. It indicates that the system performance can be improved through the cooperation of the MDs.

VI. Conclusion

In this article, we studied the joint task partitioning and power control problem in a fog computing network with multiple MDs and FDs, where each MD is energy-constrained and has a delay-sensitive computation task that needs to be executed within a system time slot. To meet the delay and energy consumption constraints, the task of each MD can be partitioned into subtasks and are collaboratively executed by the MD and the FDs. To tackle this problem, we proposed an MADDPG-based task offloading algorithm to decide the task partition strategy and transmission power strategy for each MD to maximize the long-term system utility concerning the execution delay and energy consumption. Based on extensive simulations, we demonstrated the effectiveness of the proposed algorithm under different input task sizes, the number of FDs and MDs compared with other baseline schemes. In our future work, we want to give an in-depth analysis of this problem, and to overcome the instability during the training process when the number of MD is large by introducing the attention mechanism. Moreover, some state-of-the-art techniques like federated reinforcement learning can be applied to reduce the information sharing cost and protect privacy.

References

[1] Z. Cheng, M. Min, Z. Gao, and L. Huang, "Joint task offloading and resource allocation for mobile edge computing in ultra-dense network," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Taipei, Taiwan, Dec. 2020, pp. 1–6.

[2] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6.

[3] M. LiWang, S. Dai, Z. Gao, X. Du, M. Guizani, and H. Dai, "A computation offloading incentive mechanism with delay and cost constraints under 5G satellite-ground IoV architecture," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 124–132, Aug. 2019.

[4] T. Chiu, A. Pang, W. Chung, and J. Zhang, "Latency-driven fog cooperation approach in fog radio access networks," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 698–711, Sep./Oct. 2019.

[5] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[6] Z. Ning, P. Dong, X. Kong, and F. Xia, "A Cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[7] M. Sheng, Y. Wang, X. Wang, and J. Li, "Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1524–1537, Mar. 2020.

[8] A. A. Al-Habob, A. Ibrahim, O. A. Dobre, and A. G. Armada, "Collision-free sequential task offloading for mobile edge computing," *IEEE Commun. Lett.*, vol. 24, no. 1, pp. 71–75, Jan. 2020.

[9] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, vol. 6, pp. 12825–12837, 2018.

[10] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019.

[11] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber-wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, May 2018.

[12] S. Zhou and W. Jadoon, "The partial computation offloading strategy based on game theory for multi-user in mobile edge computing environment," *Comput. Netw.*, vol. 178, no. 5, 2020, Art. no. 107334.

[13] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[14] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[15] J. Yan, S. Bi, and Y. J. A. Zhang, "Offloading and resource allocation with general task graph in mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5404–5419, Aug. 2020.

[16] L. Xiao, X. Lu, T. Xu, X. Wan, W. Ji, and Y. Zhang, "Reinforcement learning-based mobile offloading for edge computing against jamming and interference," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6114–6126, Oct. 2020.

[17] M. Min *et al.*, "Learning-based privacy-aware offloading for healthcare IoT with energy harvesting," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4307–4316, Jun. 2019.

[18] M. Hu, L. Zhuang, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Learning driven computation offloading for asymmetrically informed edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1802–1815, Aug. 2019.

[19] J. Baek and G. Kaddoum, "Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1041–1056, Jan. 2021.

[20] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5688–5698, Aug. 2021.

[21] J. Heydari, V. Ganapathy, and M. Shah, "Dynamic task offloading in multi-agent mobile edge computing networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[22] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Dec. 2017, pp. 6379–6390.

[23] K. Guo, M. Sheng, T. Q. S. Quek, and Z. Qiu, "Task offloading and scheduling in fog RAN: A parallel communication and computation perspective," *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 215–218, Feb. 2020.

[24] J. Cao, L. Yang, and J. Cao, "Revisiting computation partitioning in future 5G-based edge computing environments," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2427–2438, Apr. 2019.

[25] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 883–893, Jun. 2020.

[26] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6201–6213, Jul. 2020.

[27] X. Wang, Z. Ning, and S. Guo, "Multi-agent imitation learning for pervasive edge computing: A decentralized computation offloading algorithm," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 411–425, Feb. 2021.

[28] Z. Liu, Y. Yang, K. Wang, Z. Shao, and J. Zhang, "POST: Parallel offloading of splittable tasks in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3170–3183, Apr. 2020.

[29] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[30] P. Cheridito, H. Kawaguchi, and M. Maejima, "Fractional ornstein-uhlenbeck processes," *Electron. J. Probab.*, vol. 8, no. 3, pp. 1–14, Jan. 2003.

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[32] Y. Al-Eryani, M. Akrout, and E. Hossain, "Multiple access in cell-free networks: Outage performance, dynamic clustering, and deep reinforcement learning-based design," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 4, pp. 1028–1042, Apr. 2021.

[33] A. Gao, T. Geng, S. X. Ng, and W. Liang, "A continuous policy learning approach for hybrid offloading in backscatter communication," *IEEE Commun. Lett.*, vol. 25, no. 2, pp. 523–527, Feb. 2021.

[34] Y. Zhang, Z. Mou, F. Gao, J. Jiang, R. Ding, and Z. Han, "UAV-enabled secure communications by multi-agent deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 11599–11611, Oct. 2020.
[35] Q. Jing *et al.*, "Quantifying the performance of federated transfer learning," 2019. [Online]. Available: arXiv:1912.12795.
[36] J. Hu and M. P. Wellman, "Nash *Q*-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Dec. 2003.
[37] Y. Wang, B. Han, T. Wang, H. Dong, and C. Zhang, "Off-policy multi-agent decomposed policy gradients," 2020. [Online] Available: arXiv:2007.12322.

Zhipeng Cheng received the B.S. degree in communication engineering from Jiangnan University, Wuxi, China, in 2017. He is currently pursuing the Ph.D. degree in communication engineering with Xiamen University, Xiamen, China.

His research interests include wireless communication, task offloading, reinforcement learning, and federated learning.

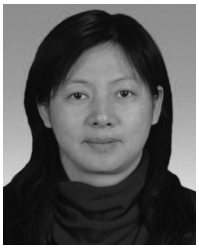
Minghui Min received the B.S. degree in automation from Qufu Normal University, Rizhao, China, in 2013, the M.S. degree in control theory and control engineering from Shenyang Ligong University, joint training with Shenyang Institute of Automation, Chinese Academy of Sciences, Shengyang, China, in 2016, and the Ph.D. degree from the Department of Information and Communication Engineering, Xiamen University, Xiamen, China, in 2019.

She was a Visiting Scholar with the Department of Computer Science and Engineering, University of Houston, Houston, TX, USA. She is currently a Teacher with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China. Her research interests include network security, privacy, and wireless communications.

Minghui Liwang (Member, IEEE) received the Ph.D. degree from the School of Informatics, Xiamen University, Xiamen, China, in 2019.

She is currently a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON, Canada. She was a Visiting Scholar with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA, from 2017 to 2018. Her research interests include wireless communication systems, Internet of Things, cloud/edge/service computing, as well as economic models and applications in wireless communication networks.

Dr. Liwang has served as the TPC Member of Communication QoS, Reliability and Modeling, held in conjunction with IEEE GLOBECOM 2020.



Lianfen Huang received the B.S. degree in radio physics and the Ph.D. degree in communication engineering from Xiamen University, Xiamen, China, in 1984 and 2008, respectively.

She was a Visiting Scholar with Tsinghua University, Beijing, China, in 1997. She is a Professor with the Department of Communication Engineering, Xiamen University. Her current research interests include wireless communication, wireless network, and signal processing.

Zhibin Gao (Member, IEEE) received the B.S. degree in communication engineering, the M.S. degree in radio physics, and the Ph.D. degree in communication engineering from Xiamen University, Xiamen, China, in 2003, 2006, and 2011, respectively.

He is a Senior Engineer of Communication Engineering with Xiamen University. His current research interests include wireless communication, mobile network resource management, and signal processing.