Multi-Agent DRL for Task Offloading and Resource Allocation in Multi-UAV Enabled IoT Edge Network

Abegaz Mohammed Seid, Gordon Owusu Boateng, Bruce Mareri, Guolin Sun, *Member, IEEE*, and Wei Jiang, *Senior Member, IEEE*

***Abstract*—The Internet of Things (IoT) edge network has connected lots of heterogeneous smart devices, thanks to unmanned aerial vehicles (UAVs) and their groundbreaking emerging applications. Limited computational capacity and energy availability have been major factors hindering the performance of edge user equipment (UE) and IoT devices in IoT edge networks. Besides, the edge base station (BS) with the computation server is allowed massive traffic and is vulnerable to disasters. The UAV is a promising technology that provides aerial base stations (ABSs) to assist the edge network in enhancing the ground network performance, extending network coverage, and offloading computationally intensive tasks from UEs or IoT devices. In this paper, we deploy a clustered multi-UAV to provide computing task offloading and resource allocation services to IoT devices. We propose a multi-agent deep reinforcement learning (MADRL)-based approach to minimize the overall network computation cost while ensuring the quality of service (QoS) requirements of IoT devices or UEs in the IoT network. We formulate our problem as a natural extension of the Markov decision process (MDP) concerning stochastic game, to minimize the long-term computation cost in terms of energy and delay. We consider the stochastic time-varying UAVs' channel strength and dynamic resource requests to obtain optimal resource allocation policies and computation offloading in aerial to ground (A2G) network infrastructure. Simulation results show that our proposed MADRL method reduces the average costs by 38.643%, and 55.621% and increases the reward by 58.289% and 85.289% compared with the different single agent DRL and heuristic schemes, respectively.**

***Index Terms*—Computation offloading, MADRL, massive IoT, multi-UAV, resource allocation.**

Manuscript received October 22, 2020; revised June 19, 2021; accepted July 3, 2021. Date of publication July 12, 2021; date of current version December 9, 2021. This work is supported by the National Natural Science Research Foundation of China, Grant No. 61771098, by the Fundamental Research Funds for the Central Universities under Grant No. ZYGX2018J068, and by the fund from the Department of Science and Technology of Sichuan province, Grant No.2020YFQ0025. The associate editor coordinating the review of this article and approving it for publication was J. S. Silva. *(Corresponding author: Guolin Sun.)*

Abegaz Mohammed Seid, Gordon Owusu Boateng, Bruce Mareri, and Guolin Sun are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: guolin.sun@uestc.edu.cn).

Wei Jiang is with the German Research Center for Artificial Intelligence (DFKI GmbH), 67663 Kaiserslautern, Germany, and also with the Department of Electrical and Information Technology, Technische University, 67663 Kaiserslautern, Germany.

Digital Object Identifier 10.1109/TNSM.2021.3096673

I. Introduction

THE CURRENT fifth generation (5G) and beyond 5G (B5G) technology is envisioned to improve the quality of service (QoS), increase productivity and network capacity of Internet of things (IoT) networks. This will reduce computation costs by using current network technology and machine learning approaches. The integration of IoT and machine learning is one of the promising paradigms, that will accelerate future network technology and increase IoT reusability in different sectors [1]. In the current network technology in 5G and B5G, the number of ultra-dense heterogeneous IoT devices at the edge network is increasing exponentially [2]. IoT devices generate a massive amount of traffic while due to a lack of computing resources, battery life, and network coverage, they cannot process all data on their own [3].

Mobile edge computing (MEC) [4] is a promising paradigm to improve the QoS of edge devices and network performance better than cloud computing infrastructure. The IoT devices can offload computation-intensive data with ultra-low latency (ULL) and low power consumption to the nearest MEC. However, the MEC base station (BS) with computing servers may be overloaded by processing massive traffic from IoT devices as a result of UE mobility, damaged due to inevitable natural disasters or temporary malfunctioning. Due to this, the traditional MEC technology is not suitable to ensure the required QoS, network availability and scalability. Therefore, the recent UAV technology has become part of MEC in a cellular communication environment to assist the ground network, acting as a macro BS (MBS) or aerial BS (ABS) [5].

Recently, the integration of UAVs into the IoT network is applicable in different application areas. Motlagh *et al.* [6] presented a comprehensive survey on the potential merits of UAV-based IoT services and challenges. Authors in [7], [8] presented the combination of UAVs and IoT devices to improve IoT services. The work in [8] discussed the synergy between UAVs and IoT systems, referred to as Internet of UAVs (IoUAVs), where UAVs, IoT devices, and humans interact in a cooperative manner. Research works have been done on UAV-assisted terrestrial devices to improve QoS in aerial to ground (A2G) networks. In [7], [9], [10] the UAV-assisted edge network, with problems concerned with UAVs controlled by the ground BSs, were solved using conventional optimization techniques to minimize energy consumption.

Software-defined networking (SDN) technology [11] is used to separate the control plane from the data plane and to increase the control mechanism. To reduce the network complexity in the A2G network environment, the SDN controller plays a vital role in managing the network. The UAV network with the SDN paradigm is a better alternative to reduce complexity, provide flexible services, and separate control from UAVs and programmability [12], [13]. Recently, reinforcement learning (RL) algorithm have been widely used in a wireless network to optimize different problems and improve the system's performance while increasing the intelligence of edge devices. Works on MEC [14]–[17] proposed a single-agent deep reinforcement learning (DRL)-based computation offloading and resource allocation frameworks to obtain optimal offloading policy and efficient resource allocation in a dynamic environment. In the A2G network, different DRL algorithms [18]–[22] are adopted to either minimize computation cost or maximize the network utility.

When there is an intensive task of IoT devices offloading onto the MEC server or UAV network, the main challenge is to obtain the optimal offloading policy by heuristic or learning techniques, and choosing the computational node to offload tasks in a given time slot parallelly [14]. Since a local MEC server has limited capacity, it cannot serve a massive number of requests from multiple agents simultaneously, and the network traffic is also critical. The association is too complicated, and the waiting time is high. Because of this, the computation costs also increases, and the IoT device may drop tasks due to low power or under QoS thresholds. Due to this situation, the local computation perhaps performs better than offloading tasks to computational nodes. Besides, the local BS with the MEC server may be malfunctioning in a specific network area [23], [24].

The multi-agent reinforcement learning (MARL) approach can resolve the current real-world (multiple agent environment) IoT edge network problems. It provides distributed resource management for computation offloading with resource allocation efficiently at IoT edge networks; each IoT device can take action according to its observed local information. MADRL has been applied in traditional MEC network in different problems such as handling risk-aware energy [25], and channel access and computation offloading [26]. In UAV network, MADRL methods have been applied to optimize simultaneous target assist and path planning (STAP) [27], dynamic resource allocation to ground users in UAV network [28], [29]. These works formulated a problem using stochastic game and applied MADRL algorithm to solve a formulated problem. However, these works are focused either on MEC network or UAV network and do not consider the network congestion and emergency in UAV assisted edge IoT network scenario. Based on our knowledge and the limitation of the [25]–[29] works, we propose a multi-UAV enabled IoT edge network and multi-agent DRL approach to obtain an optimal offloading policy decision based on local observation and past information. We consider the computation costs in terms of computation delays and energy consumption in a multi-UAV enabled IoT edge network. We propose a multi-agent DRL framework to solve the IoT edge network problem in the A2G network environment. The multiple agents can cooperate to reduce the computation cost and increase the association (offloading) probability rate within computational nodes based on their prior knowledge of the system parameters. In particular, our main contributions of this work are as follows.

1) We design a multi-UAV enabled IoT edge network for dynamic task offloading and resource allocation in a cooperative environment.
2) We propose an MADRL-based method for modeling the cooperative computation offloading, resource allocation, and association problem to provide services for different IoT devices at IoT edge network.
3) We formulate the above-mentioned problem as a stochastic game where we assume each agent makes a decision according to local real-time observations and current strategies to select a resource, choose task offloading, and also share information with other agents.
4) By adopting the multi-agent deep deterministic policy gradient (MADDPG) method, the multiple agents in the A2G network environment can maximize the long-term reward to reduce the computation cost and effective resource allocation by finding optimal policy and reducing the overall training cost by centralized training and decentralized execution technique.

The rest of the paper is organized as follows: In Section II, we review related works. Section III introduces the proposed system model. In Section IV, we introduce the MADRL scheme in our problem and stochastic game based problem formulation. Discuss the MADDPG based solution in Section V. Section VI presents the simulation results and analysis. Finally, we conclude this work in Section VII.

II. Related Works

In 5G and B5G technology, UAVs are one of the MEC system equipment that act as UEs and can access the MEC server for different services or act as ABSs to provide services to ground IoT devices. The authors in [30], [31] studied how to replace destroyed BSs with ABSs after creating multi-hop D2D communication. Further extending the wireless coverage to guarantee the QoS of UAV communications for IoT network in disasters and relaying the signal for emergency communication vehicles was discussed.

The traditional single-agent DRL approach is not well applicable in multi-agent systems and the current IoT edge network. Multi-agent DRL frameworks like MADDPG, consider the action policies of other agents during training [32] and the agents learn in cooperative strategies. The MADRL benefit in the MADDPG algorithm is applicable for centralized training with a decentralized execution method. Recently, researchers have capitalized on MARL algorithms to tackle different multi-agent system problems in wireless cellular networks. Some of the works have been done in [25]–[29], [33]–[42], which adopted the MARL approach for computation offloading and resource allocation in different scenarios.

Yan *et al.* [37], [38] proposed an MARL-based smart multi-RAT access in a heterogeneous network (HetNet) to

TABLE I
SUMMARY OF RELATED WORK

Paper	Scenario	Method	Problem	Objective	Agent
[25]	MEC network	A3C	Risk aware energy scheduling	Reduce loss of energy	Virtual agent
[26]	MEC-enabled industry 4.0	MADDPG	Channel allocation	Max. channel gain	MTA
[27]	UAV network	MADDPG	STAP	Collision avoidance	UAV
[28], [29]	UAV network	Q-learning	power and channel allocation	Maximize throughput	UAV
[33]	Cognitive radio (CR)	Q-learning, SARSA	Power allocation	Max. energy efficiency	CR-BS
[34]	NOMA assisted MA-MEC	DNN	Computation offloading NOMA transmission Computation resource allocation	Min. total energy cons. of IoTD	IoTD
[36]	MEC network	LSTM	Energy consumption	Min. total energy cost of system	BS
[37], [38]	Radio access technology	Q-learning	Resource allocation	Max. network throughput	UE
[39]	Internet of Vehicle (IoV)	MADDPG	Control traffic load	Min. control delay	Edge controller
[41]	IoT Network	IL based MA-Q	Spectrum and computation res.	Min. system cost	User
[42]	UAV network	Q-learning	Spectrum sharing	Optimal task allocation	UAV

maximize the network resource utilization through maximizing users' long-term network throughput and ensuring the QoS of users.

The authors in [40] discussed the merits of the MARL approach in a dynamic vehicular network to obtain a reliable solution for different problems. The work in [41], proposed an MARL-based computation offloading with a resource allocation (selecting transmission power level, subchannel, and radio access) problem in an IoT network and formulated the problem using a stochastic game. The objective was to minimize long term system costs. In [26], the authors presented an MADRL approach in MEC enabled Industry 4.0 IoT network to solve computation offloading and multiple channel access problems. The agents maximize the long-term reward by reducing the delay and improving channel access rate. The problem was formulated and solved using a stochastic game. A2G networks have also adopted MARL approaches for different problems [27]–[29], [42]. Some authors in [28], [29] presented an MARL-based dynamic resource allocation in multi-UAV networks that provide services to users. The problem was formulated using stochastic game theory and solved by a multi-agent Q-learning approach, i.e., each UAV makes a decision according to its observed information, to select the transmission power level and subchannel to provide services for the user at the ground network.

The above-mentioned works have been done on the computation offloading and resource allocation problem based on the MARL approach. However, the focus was on the ground edge network equipment and concentrated more on channel or spectrum allocation and power allocation in a multi-agent environment, which considered the agents' resources only. A single agent cannot handle the highly dimensional environment due to large state-action space, and the convergence rate requires many iterations.

Based on the delay and resource constraints of IoT applications and limitations of existing works, we investigate a multi-UAV enabled IoT edge network for computation offloading, and resource allocation, which has not been well-tackled in the previous works. We are motivated by the application of MARL in the A2G network focused on multi-UAV enabled IoT edge network computation offloading and resource allocation. TABLE I provides a comparative summary of some closely-related works.

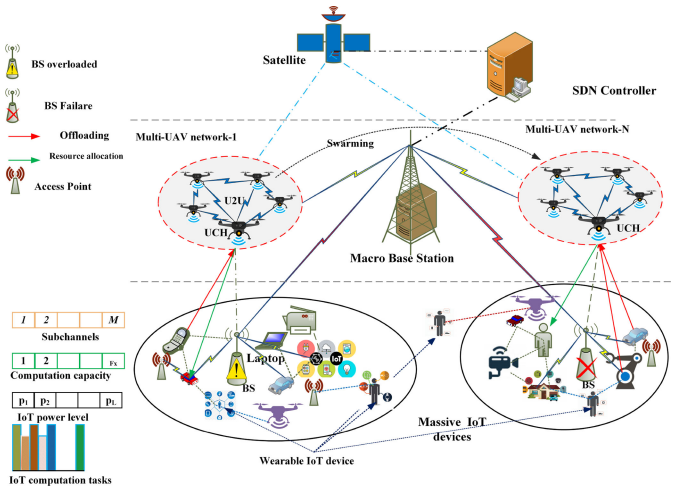


Fig. 1. Multi-UAV enabled IoT edge network system model.

III. SYSTEM MODEL

In multi-UAV enabled IoT edge network, several network operations and traffics are processed. Due to this, the network design becomes complicated. However, the programmable SDN controller can manage network complexity in an easier way. In our scenario, a single centralized SDN controller manages the A2G network using standard OpenFlow communication protocols [43].

Fig. 1 depicts the proposed multi-UAV enabled IoT edge network system model. We consider a central network controller (SDN), one MBS with a MEC server, a UAV cluster network swarm over K small cells consisting of L smart edge devices (IoT devices), and access points. We consider a multi-UAV, multi-cell, and multi-computational node (UAV cluster head (UCH) as an aerial MEC server, a BS with local MEC) to provide different services to heterogeneous IoT devices. Each local BS with a local MEC server and UCH provides resource allocation and computation offloading services to edge IoT devices in each cell.

A. Network Model

Based on 5G communication technology, we assume each edge IoT device is connected in D2D communication mode in each cell. The UAVs are also connected to each other in

TABLE II
LIST OF KEY NOTATIONS

Notation	Description
$\mathcal{L}$	Set of IoT devices
$\mathcal{U}$	Set of UCHs
$\mathcal{N}$	Set of BSs
$\mathcal{X}$	Set of active computation nodes
$\mathcal{M}$	Set of subchannels
$S_x^m(t)$	A state of subchannel between IoT device and node x at time slot t
$\alpha_{lx}^m(t)$	Task offloading decision variable at time slot t
$\mathcal{L}_{off}^x$	Set of IoT devices that offload task to node x.
$p_l(t)$	A transmission power of IoT device l at time slot t
$\gamma_{lx}^m(t)$	The SINR between IoT device l and computational node x(BS or UCH).
$s_l(t)$	State observed by IoT device l at time slot t
$a_l(t)$	Action taken by IoT device l at time slot t
$r_l(t)$	Reward obtained by IoT device l at time slot t
$s' = s(t+1)$	Transition state
$\bar{\gamma}_{lx}(t)$	State of SINR between IoT device l and node x at time slot t
$\mathcal{T}_l(t)$	State of IoT device l task information at time slot t
$\mathcal{P}_l(t)$	State of power level of IoT device l at time slot t
$\nu_u(t)$	State of UCH u status at time slot t

each cluster in D2D communication fashion. Due to this, the bandwidth resources are utilized efficiently and transmission power is reduced by minimizing the distance between edge IoT devices and BSs and increasing the computation capacity of the UAV network by computing tasks in a distributed manner.

The ground computational node serves a massive number of heterogeneous IoT devices. Some of the IoT devices (sensors on the vehicle, body sensors, wearable devices) are movable. Perhaps, those devices may be far from the network coverage, and malfunctioning may exist at a section of the ground network, causing network performance degradation. Therefore, we deploy the UAV flying BS to enable the IoT edge network. The multi-UAV network is organized in clusters, which swarm over the ground edge network and accomplish their mission-critical tasks. The UAVs also act as aerial MEC servers and provide different services to ground network equipment. We denote a set of IoT devices as $l \in \mathcal{L} = \{1, 2, \ldots, L\}$, a set of UCHs $u \in \mathcal{U} = \{1, 2, \ldots, U\}$, a set of BSs $n \in \mathcal{N} = \{1, 2, \ldots, N\}$ and a set of active computational nodes[1] either UCH or BS MEC server is $x \in \mathcal{X} = \{1, 2, \ldots, X\}$. When an IoT device *l* chooses to offload the tasks onto the BS or UCH with access channel *m*, the uplink rate is expressed as:

$$r_{l,x}^{uplink} = Blog_2\Big(1 + \frac{p_l g_{lx}}{\sigma^2}\Big), \tag{1}$$

where g_{lx} is the channel gain of IoT device l to computational node x on channel m; p_l is the transmission power of IoT device l, and σ^2 is the system noise power. Each subchannel has two states at each time slot t, either available or disconnected and is expressed as $s_x^m(t) \in \{0, 1\}$. If $s_x^m(t) = 1$, the subchannel of BS or UCH is available, and the IoT device can offload the task; else, the subchannel is unavailable, which means either the computation node is overloaded or broken down. The number of IoT devices associated with either one of UCH or BS x is limited and can be expressed as $\sum_{l=1}^{\mathbf{L}} \leq {}_x, \forall x \in \mathcal{X}$, where ${}_x$ denotes the maximum allowed number of IoT devices whose tasks are offloaded and associated with either UCH *u* or BS *n*. We consider that each IoT device can only select one computational node using allocated spectrum or channel *m* at any time slot *t* and is expressed as:

$$\sum_{x=0}^{x-1} \alpha_{lx}^m(t) \leq 1, \ \forall l \in \mathcal{L}, m \in \mathcal{M}, \tag{2}$$

where $\alpha_{lx}^m(t)$ is the task offloading decision variable.

1) Aerial-to-Ground Communication: The A2G communication channel depends on the altitude, angle of elevation, and type of propagation environment [44]. The A2G communication links can be modeled by a probabilistic path loss (*PL*) model containing the line of sight (LoS) and non-LoS (NLoS) in different probabilities of occurrence. Based on [45], the LoS connection probability between UCH *u* and IoT device *l* at time slot *t* is expressed as:

$$P^{Los} = \frac{1}{1 + \iota exp\Big(-\varrho sin^{-1}1\Big(\frac{H_u}{D_{lu}(t)}\Big) - \iota\Big)}, \tag{3}$$

where ι and ϱ are constants that depend on the environment, $D_{lu}(t)$ represents the distance between UCH *u* and IoT device *l*, and H_u represents the altitude of UCH *u*. Therefore, the NLoS link probability is calculated as $P^{NLoS}(t) = 1 - p^{LoS}(t)$. Based on the above analysis, the pathloss from UCH *u* to ground IoT device *l* is calculated as:

$$\begin{aligned} PL_{u,l}^{LoS} &= \mathtt{L}_{u,l}^{FS} + \eta^{LoS}, \\ PL_{u,l}^{NLoS} &= \mathtt{L}_{u,l}^{FS} + \eta^{NLoS}, \end{aligned} \tag{4}$$

where $\mathtt{L}_{u,l}^{FS}$ represents the free space pathloss with $\mathtt{L}_{u,l}^{FS} = 20log(D_{lu}(t)) + 20log(f) + 20log(\frac{4\pi}{\bar{c}})$, f is the frequency carrier, and $\bar{c}$ is the speed of light. Then, the average pathloss between UCH *u* and IoT device *l* at time slot *t* is expressed as:

$$\bar{\mathtt{L}}_{u,l}(t) = P^{LoS}(t).PL_{u,l}^{LoS}(t) + P^{NLoS}(t).PL_{u,l}^{NLoS}(t). \tag{5}$$

B. Computation Model

Each edge IoT device has different forms of applications, and the size of tasks are generated at time slot *t*. Each computation task has three profiles: the size of the input task (d_l), the required computation capacity (number of CPU cycles) processing task c_l, and the maximum time delay $\bar{T}_l$. The task profile ($\mathbb{T}_l(t)$) of IoT device *l* is expressed as $\mathbb{T}_l(t) = \{d_l(t), c_l(t), \bar{T}_l(t)\}, l \in \mathcal{L}$.

1) Local Computation: For each IoT device *l*, the input task $\mathbb{T}_l(t)$ is either computed locally or is offloaded onto computational node *x*, which computes either on local BS's MEC server or UCH at time slot *t*. When an IoT device decides to compute a task locally ($\alpha_{lx}^m(t) = 0$), the completion time $\mathbb{T}_l(t)$ is calculated as:

$$T_l^{loc}(t) = \frac{c_l(t)}{f_l^{loc}(t)} \leq \bar{T}_l, \quad \forall l \in \mathcal{L}, \tag{6}$$

where $f_l^{loc}(t)$ is the computation capacity (CPU cycles) of the IoT device. The energy consumption during local task execution at IoT device *l* is calculated as $E_l^{loc}(t) = \kappa_l(f_l^{loc}(t))^2$,

[1] $n, u \subset x$.

where κ_l represents the energy coefficient of CPU [46]. Let us denote ω^t and ω^e as weighted parameters of time delay and energy consumption, respectively. The local computation cost of the task $\mathbb{T}_l(t)$ of IoT device l is expressed as: $\Omega_l^{loc}(t) = \omega^t T_l^{loc}(t) + \omega^e E_l^{loc}(t)$.

2) IoT Task Offloading Model: When the IoT device decides to offload tasks onto the associated computational node based on the current policy and other IoT devices' information, the time delay cost is calculated in three phases: transmission time, execution time, and time delay sending the result back.

We assume the A2G system has multiple channel access schemes in orthogonal frequency-division multiple access (OFDMA) technique. Let $\mathcal{W}$ be an operational frequency band that can be divided into an equal subchannel $B = \frac{\mathcal{W}}{M}[Hz]$. Each computational node can serve at most *L* IoT devices at time slot *t*. In each computational node, the set of available subchannels is defined as $m \in \mathcal{M} = \{1, 2, 3, \ldots, M\}$. Let us define the task offloading decision variable at time slot *t* with uplink subchannel scheduling as $\alpha_{lx}^m(t) = \{-1, 0, 1\}, l \in \mathcal{L}, x \in \mathcal{X}, m \in \mathcal{M}$, where $\alpha_{lx}^m(t) = 1$ indicates that the IoT device l decides to offload input task $\mathbb{T}_l$ onto BS *n*, $\alpha_{lx}^m(t) = -1$ indicates that the IoT device *l* decides to offload input task $\mathbb{T}_l$ onto UCH *u* on subchannel *m*, and $\alpha_{lx}^m(t) = 0$ indicates the task is computed by the IoT device itself. The IoT device *l* defines the task offloading strategy based on current policy at time slot *t* as $\bar{A} = \{\alpha_{lx}^m(t) \in O(t)\} | \alpha_{lx}^m(t) = \{-1, 1\}, l \in \mathcal{L}, x \in \mathcal{X}$, where $O(t)$ is a variable that contains all IoT devices' task offloading decisions and can be expressed as $O(t) = \{\alpha_{lx}^m(t)\} | l \in \mathcal{L}, x \in \mathcal{X}, m \in \mathcal{M}$. Every IoT device *l* with current policy must satisfy the following expression:

$$\sum_{x \in \mathcal{X}, m \in \mathcal{M}} \alpha_{lx}^m \leq \{-1, 0, 1\}. \tag{7}$$

When the IoT device l offloads input tasks onto a computational node, it has also calculated the transmission power for uploading the tasks. We define the set of IoT devices offloading intensive tasks onto BS or UCH as: $L_{off}^{(x)} = \{\sum_{m \in \mathcal{M}} \alpha_{lx}^m(t) = 1, |-1|\}, l \in \mathcal{L}, x \in \mathcal{X}, m \in \mathcal{M}$. The transmission power of an IoT device l uploading input task d_l onto either BS *n* or UCH *u* is denoted as $p_l(t)$, where $\mathcal{P} = \{p_l(t) | 0 \leq p_l(t) \leq p_l^{max}, l \in L_{off}^x\}$, according to the current policy at time slot *t*. During uploading/transmission of input tasks onto the nearest computational node accessing different subchannels, the IoT device suffers from inter-cell interference. Therefore, we define the Signal-to-Interference-Plus-Noise-Ratio (SINR) from IoT device *l* onto computational node *x* on subchannel *m* depending on the following cases at time slot *t*;

Case-1: If the IoT device *l* chooses to associate with BS *n* ($\alpha_{lx}^m(t) = 1$), the SINR is expressed as:

$$\gamma_{ln}^m(t) = \frac{p_l(t) h_{ln}^m(t)}{\sum_{j \in \mathcal{N} \neq \{n\}} \sum_{i \in \mathcal{L}_j} \alpha_{iju}^m(t) p_j(t) h_{ij}^m(t) + \sigma^2}, \tag{8}$$

where $h_{ln}^m(t)$ is the channel gain between IoT device *l* and BS *n* on subchannel *m*, and σ^2 is the power of noise variance. According to the above SINR, the achievable data rate of IoT device *l* offloading task onto BS *n* is calculated as:

$$r_{ln}(t) = B log_2(1 + \gamma_{ln}), \tag{9}$$

where $\gamma_{ln} = \sum_{m \in \mathcal{M}} \gamma_{ln}^m$. Let γ' denote the targeted QoS threshold of IoT devices served by a computational node. If $\alpha_{lx}^m(t) = 1$, the input task d_l transmission delay cost of IoT device *l* according to the current policy is calculated as:

$$T_l^{tr}(t) = \sum_{n \in \mathcal{N}} \alpha_{lx}^m(t) \left(\frac{d_l(t)}{r_{ln}(t)} \right), \ \forall l \in \mathcal{L}. \tag{10}$$

Case 2: If the IoT device receives a lousy signal or information about subchannel and overloaded (failed) BS *n* from another IoT device, it also chooses the aerial BS (ABS). As defined above ($\alpha_{lx}^m(t) = -1$), the IoT device *l* is served by UCH *u* in time slot *t*. Let us define the SINR between IoT device *l* and UCH *u* over subchannel *m* in time slot *t* as:

$$\gamma_{lu}^m(t) = \frac{G_{lu}^m(t) \alpha_{lx}^m(t) h_u^m(t) p_u(t)}{I_{lu}^m(t) + \sigma^2}, \tag{11}$$

where $G_{lu}^m(t)$ is the channel gain between UCH *u* and IoT device *l* over subchannel *m*, $p_u(t)$ is the transmission power of UCH *u*, $I_{lu}^m(t)$ is the interference between UCH *u* and another UCH *j*, $I_{lu}^m(t) = \sum_{(i \in \mathcal{U}, i \neq u)} G_{li}^m(t) h_u^m(t) p_i(t)$. The SINR of UCH *u* in any time slot *t* is expressed as $\gamma_{lu} = \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} \gamma_{lu}^m$. We define the coordinate of IoT device *l* in time slot *t* as $(x_l(t), y_l(t), 0)$ and the coordinates of UCH *u* is also represented as $(X_u(t), Y_u(t), H_u(t))$. Then, the horizontal distance in time slot *t* between IoT device *l* and UCH *u* is calculated as:

$$D_{lu}(t) = \sqrt{(X_u(t) - x_l(t))^2 + (Y_u(t) - y_l(t))^2}, \quad \forall l \in \mathcal{L}, u \in \mathcal{U}. \tag{12}$$

The channel gain between IoT device *l* and UCH *u* is calculated as $g_{lu}(t) = \frac{g_o}{(H_u)^2 + D_{lu}(t)}$, where g_o denotes the channel power gain at the reference distance of 1m, and H_u is the altitude of UCH *u*. Even if the channel strength or SINR of UCH is available, the IoT device *l* also checks the UAV network coverage $D_{lu}(t) \leq H_u(t) tan\theta_u$ before offloading the task, where θ is the angle of elevation. According to the above explanation, the offloading data rate can be calculated as:

$$r_{lu}(t) = B log_2 \left(1 + \frac{\varphi p_{lu}(t)}{\theta_u^2 (H_u^2 + D_{lu}(t))^2} \right), \ \forall l \in \mathcal{L}, \tag{13}$$

where *B* represents the subchannel bandwidth, $\varphi = \frac{g_o(G_o)}{\sigma^2}$, $G_0 \approx 2.2846$, $p_{lu}(t)$ is the transmission power of IoT device *l* in time slot *t*, and σ^2 the noise power. If $\alpha_{lx}^m(t) = -1$, the offloading transmission delay between IoT device *l* and UCH *u* based on the current policy in time slot *t* is calculated as:

$$T_{lu}^{tr}(t) = |\alpha_{lx}^m(t) \left(\frac{d_l(t)}{r_{lu}(t)} \right)|^2, \ \forall l \in \mathcal{L}. \tag{14}$$

The execution time delay of offloaded task from IoT device *l* onto the computational nodes based on the current policy and other IoT devices' information in time slot *t* will be calculated based on the computation resource block of the local MEC server and UCH capacity. Each resource block on the

computational node is shared by all IoT devices at time slot *t*. The computation resource capacity of local MEC at BS *n* is represented as f_n, and UCH capacity is f_u. We assume that the computation resource block is allocated equally for all IoT devices. The computation resource allocation policy is defined as $F_n = \{f_{ln}(t)|l \in \mathcal{L}, n \in \mathcal{N}\}, F_u = \{f_{lu}(t)|l \in \mathcal{L}, u \in \mathcal{U}\}$. The computation resource allocated by BS server or UCH to IoT device l must be greater than zero in time slot *t*. If $f_{ln}(t)|f_{lu}(t) = 0$, the computation resource is not available. The computation resource constraint must satisfy the following expression; $\sum_{l\in\mathcal{L}} f_{ln}(t) \leq f_n, \forall n \in \mathcal{N}$, and $\sum_{l\in\mathcal{L}} f_{lu}(t) \leq f_u, u \in \mathcal{U}$. The local MEC server at BS n or UCH *u* will complete the offloaded task with a delay cost calculated as follows:

$$T_{lx}^{exe}(t) = \begin{cases} \sum_{n\in\mathcal{N}} \alpha_{lx}^{m}(t)\frac{c_l(t)}{f_{ln}(t)}, & Case\text{-}1 \\ \sum_{u\in\mathcal{U}} \left|\alpha_{lx}^{m}(t)\frac{c_l(t)}{f_{lu}(t)}\right|, & Case\text{-}2 \end{cases} \\ \forall l \in \mathcal{L}, \forall n, \forall u \in \mathcal{X}. \tag{15}$$

Therefore, the total delay cost experienced by IoT device l according to the current policy $T_l^{tot}(t)$ is $T_l^{tot}(t) = T_l^{tr}(t) + T_l^{exe}(t)$. The total delay cost to compute offloaded task of IoT device l is expressed as:

$$T_{lx}^{tot}(t) = \begin{cases} \sum_{n\in\mathcal{N}} \alpha_{lx}^{m}(t)\left(\frac{d_l(t)}{r_{nl}(t)} + \frac{c_l(t)}{f_{ln}(t)}\right), & Case\text{-}1 \\ \sum_{u\in\mathcal{U}} \left|\alpha_{lx}^{m}(t)\left(\frac{d_l(t)}{r_{ul}(t)} + \frac{c_l(t)}{f_{lu}(t)}\right)\right|, & Case\text{-}2 \\ \forall l \in \mathcal{L}, \forall n \in \mathcal{X}, \forall u \in \mathcal{X} \end{cases} \tag{16}$$

For simplicity, we ignore time cost returning the result from either BS *n* or UCH *u* to IoT device l [47].

C. Energy Model

In this subsection, we discuss the energy consumption cost,[3] when the IoT device l offloads a task onto a computational node in different phases such as transmission, execution, UCH *u* flying and UCH *u* hovering energy consumption. First, we assume each IoT device *l* and UCH *u* adopts discrete transmit power control. The transmit power values of each IoT device l is represented as a vector $\mathcal{P}_l^T = \{p_1, .., p_{\mathcal{Z}}\}$. At time slot *t*, each IoT device selects its transmit power as $\mathcal{P}_{l,z}^T(t), z \in \mathcal{Z} = \{1, .., Z\}, \mathcal{P}_{l,z}^T(t) = 1$ and $\alpha_{lx}^m(t) \in \{-1, 1\}$, indicates IoT device l decides to offload the input task, otherwise, $p_{l,z}^T(t) = 0$. Based on other constraints and current policy, the IoT device selects a transmission power from UCH *u* ($p_{u,z}^T(t) = 1$) or chooses local task execution. The power resource is critical on UAVs, so the power value of each UCH to communicate with a chosen IoT device l can be represented as $\mathcal{P}_u^T = \{p_1, \ldots, p_{\mathcal{Z}}\}$. The binary values of power are denoted as $p_{u,z}^T(t), z \in \mathcal{Z} = \{1, \ldots, Z\}$. If $p_{u,z}^T(t) = 1$, it indicates the UCH *u* allocates transmit power level p_z to IoT device l in time slot *t*, otherwise, $p_{u,z}^T(t) = 0$, and there is no transmission power allocation at time slot *t*. Either the IoT device l or UCH *u* can select/allocate a power level at each time slot *t* and is expressed as:

$$\sum_{z\in\mathcal{Z}} p_{l,u,z}^T, \ u \in \mathcal{U}, l \in \mathcal{L}. \tag{17}$$

Based on the above transmission power control strategies, we can calculate the energy consumption of the input task. If the transmission power level is $p_{l,z}^T(t) = 0$, and the IoT device decides to compute the input task locally, the energy consumption during local computing is calculated as:

$$E_l^{exe}(t) = \kappa_l (f_l(t))^2, \ l \in \mathcal{L}, \kappa_l > 0. \tag{18}$$

Otherwise, the IoT device l decides to offload the task onto UCH *u* or BS *n*. The transmission power $p_{l,z}^T(t) = 1$ according to the policies, and the energy consumption is expressed as:

$$E_{lx}^{tr}(t) = \sum_{x=1}^{\mathcal{X}} |\alpha_{lx}^m(t) p_{lx}(t)|, l \in \mathcal{L}. \tag{19}$$

The power constraint of IoT device l is expressed as $\alpha_{lx}^m(t)(E_l^{exe}(t) + E_l^{tr}(t)) \leq P_l^{max}, l \in \mathcal{L}$. Based on the above cases, we can calculate the energy consumption of transmission and execution costs from IoT device l up to the computational node. When the IoT device l has chosen or associated with BS *n*, the transmission and execution power consumption is calculated as:

$$E_{ln}^{tr}(t) = P_l^{tr}(t) T_l^{tr}(t), \\ E_n^{exe}(t) = \kappa_n (f_n(t))^2, \ n \in \mathcal{N}. \tag{20}$$

The computation resource constraint at BS *n*, which is equipped with local MEC server in time slot *t* is expressed as:

$$\sum_{l=1}^{\mathcal{L}} \alpha_{lx}^m(t) f_{ln}(t) \leq F_n^{max}, \tag{21}$$

where $\alpha_{lx}^m(t) = 1, n \in \mathcal{N}$.

If the IoT device is chosen or associated with the UCH *u* according to the current policy and other IoT device information, the energy consumption is also calculated based on UCH *u* constraints such as flying, hovering, and execution energy consumption at the time slot *t*. For simplicity, we refer to flying and hovering energy consumption of UAVs interchangeably in this sequel. Let $p_u^{fl}(t)$ be UCH *u* flying power consumption and the flying energy consumption depends on flying distance and speed of UCH $V_u(t)$. Then, the flying energy consumption $E_u^{fl}(t)$ at time slot *t* is calculated as:

$$E_u^{fl}(t) = p_u^{fl}(t)\left(\frac{\sqrt{[x_u(t) - x_u(t')]^2 + [y_u(t) - y(t')]^2}}{V_u(t)}\right), \tag{22}$$

where $t' = t - 1$. The UCH *u* hovering energy consumption $E_u^{ho}(t)$ at time slot *t* is calculated as:

$$E_u^{ho}(t) = \frac{p_u^{ho}(t) d_l(t)}{r_{lu}(t)}, \tag{23}$$

[2]We use absolute value symbol due to the decision value is -1.

[3]We are more focused on UAV energy resource than BS energy because the BS energy is not much critical like UAV energy.

where, $p_u^{ho}(t)$ is the consumption during hovering at the time slot *t*. The transmission energy consumption for computation task offloading onto UCH *u* is calculated as:

$$E_{lu}^{tr}(t) = P_l^{tr}(t)\, T_l^{tr}(t). \tag{24}$$

The UCH *u* energy consumption after receiving the offloaded task from IoT device *l* at time slot *t* is expressed as:

$$E_u^{exe}(t) = \sum_{l\in\mathcal{L}} \Big| \alpha_{lx}^m(t) \kappa_u (f_u(t))^2 \Big|. \tag{25}$$

Therefore, the total energy consumption of UCH *u* in the time slot *t* calculated as:

$$E_u^{tot}(t) = \rho_1 E_u^{fl}(t) + \rho_2 E_u^{ho}(t) + \rho_3 E_u^{exe}(t), \tag{26}$$

where ρ_1, ρ_2, ρ_3 is the variable parameter for the above UCH *u* energy consumption. The computation resource constraint at UCH *u* in the time slot *t* is expressed as:

$$\sum_{l=1}^{\mathcal{L}} |\alpha_{lx}^m(t) f_{lu}(t)| \le F_u^{Max}, \tag{27}$$

where $\alpha_{lx}^m(t) = -1, u \in \mathcal{U}$. Let $\psi_u(t) \in \{0,1\}$ indicate the resource block of UCH *u* at the time *t*, where $\psi_u(t) = 1$ implies the UCH *u* resource block is available and can provide different services to IoT devices; Otherwise, $\psi_u(t) = 0$. The total energy consumption cost experienced by IoT device *l* according to the current policy is $E_l^{tot}(t) = E_l^{tr}(t) + E_l^{exe}(t)$. The total energy cost to compute offloaded task of IoT device *l* is expressed as:

$$E_{lx}^{tot}(t) = \begin{cases} \sum_{n\in\mathcal{N}} \alpha_{lx}^m(t)\big(E_{ln}^{tr}(t) + E_{ln}^{exe}\big), & \textit{Case-1} \\ \sum_{u\in\mathcal{U}} \big|\alpha_{lx}^m(t)\big(E_{lu}^{tr}(t) + E_{lu}^{exe}\big)\big|, & \textit{Case-2} \\ \forall l \in \mathcal{L}, \forall n \in \mathcal{X}, \forall u \in \mathcal{X} \end{cases} \tag{28}$$

The SDN controller is in constant communication with the UCHs instead of the individual UAVs, and IoT devices also learn from environment/experience/other IoT device experiences to reconfigure themselves. In this way, the signaling overhead and computation complexity is minimized. To control the fairness coverage of UCHs, we define the fairness level among UCHs which serve IoT devices, and among IoT devices. Let us define the fairness level $\Upsilon^u(t) \in [0,1]$ among UCHs as:

$$\Upsilon^u(t) = \frac{\Big(\sum_{u=1}^{U}\sum_{t=1}^{T} \beta^u(t)\Big)^2}{\sum_{u=1}^{U}\Big(\sum_{t=1}^{T} \beta^u(t)\Big)^2}, \tag{29}$$

where $\beta^u(t) = \frac{\sum_{l=1}^{L} \alpha_{lx}^m(t)}{L}, \beta^u(t) \in [0,1]$ indicates the relative number of IoT devices served by UCH *u* at time slot *t*. At a given time slot, all IoT devices may not be accessed by the UCHs due to different factors. To manage this situation, we define the fairness level among IoT devices $\bar{\Upsilon}^l(t) \in [0,1]$ as:

$$\bar{\Upsilon}^l(t) = \frac{\Big(\sum_{l=1}^{L}\sum_{t=1}^{T} \alpha_{lx}^m(t)\Big)^2}{\sum_{l=1}^{L}\Big(\sum_{t=1}^{T} \alpha_{lx}^m(t)\Big)^2}. \tag{30}$$

The total computation costs in terms of total time delay and energy consumption is calculated as: $\Omega_{lx}(t) = \omega^t T_{lx}^{tot}(t) + \omega^e E_{lx}^{tot}(t)$. Then the total computation cost of IoT device *l* is expressed as: $\mathcal{Z}_l(t) = \Omega_l^{loc}(t) + \Omega_{lx}(t)$, and the system cost is calculated as: $\mathcal{Z}_{lx}(t) = \sum_{l\in\mathcal{L}} \mathcal{Z}_l(t)$.

D. Cost Objective Function

We formulate the optimization problem in our scenario to minimize energy consumption , time delay and resource allocation costs, depending on decision profile $\alpha_{lx}^m(t)$, $s_x^m(t)$, fairness level of UCH *u* $\Upsilon^u(t)$ and resource of UCH *u*. Therefore, the optimization problem is expressed as follows;

$$\boldsymbol{P_1} \quad \min_{\chi,\mathcal{F},\mathcal{P}} \mathcal{Z}_{lx}(t) \tag{31a}$$

$$\text{s.t:} \quad C1: \alpha_{lx}^m(t), s_x^m(t), \psi_u(t) \in \{0,1\},\ \forall l \in \mathcal{L}, x \in \mathcal{X}. \tag{31b}$$

$$C2: \sum_{x\in\mathcal{X}} \alpha_{lx}(t) \le 1,\ \forall l \in \mathcal{L}. \tag{31c}$$

$$C3: \sum_{x\in\mathcal{X}} s_x^m(t) = 1,\ \forall l \in \mathcal{L}. \tag{31d}$$

$$C4: \sum_{x\in\mathcal{X}} f_{lx}(t) \le F_x^{max},\ \forall l \in \mathcal{L}. \tag{31e}$$

$$C5: \sum_{x\in\mathcal{X}} p_{lx}(t) \le P_x^{max},\ \forall l \in \mathcal{L}. \tag{31f}$$

$$C6: \Upsilon^u(t) \in [0,1] \tag{31g}$$

where $(\chi = \{\alpha_{lx}^m(t)\},\ \mathcal{F} = \{f_{lx}(t)\}\ ,\ \mathcal{P} = \{p_{lx}(t)\}), \forall l \in \mathcal{L}, x \in \mathcal{X}$. Here, constraint (C1) indicates the binary offloading decision strategy, subchannel availability, and resource block of UCH, respectively. Constraints (C2, C3) indicate that computation is offloaded to at most one node *x*, the subchannel is available and allocated at time slot *t*. Constraints (C4, C5) denote the maximum computation capacity and transmission power of computational node *x*, respectively. Constraint (C6) indicates fairness level of UCH at time slot *t*. Since the optimization problem above is NP-hard, we transform the problem for optimal computation offloading and efficient resource allocation into a stochastic game and solve using MADRL technique, which is discussed in the next section.

IV. MADRL for Task Offloading and Resource Allocation in A2G Network

Since the optimization problem in (31a) is complex and continuous action space problem, we transform the optimization problem into a stochastic game model to handle the complexity and dynamics in the A2G network. The main goal of the dynamic design for joint computation offloading decision and resource selection (computation resource and transmission power) is to ensure the QoS of IoT device *l*, is not less than the predefined QoS threshold as $\gamma_{lx}(t) \ge \bar{\gamma}(t)$, where $\bar{\gamma}(t)$ is predefined QoS of IoT device served by the computational node. Each IoT device at time slot *t* either computes tasks locally $(\alpha_{lx}^m(t) = 0)$ or offloads to the available computational node $(\alpha_{lx}^m(t) = 1|-1|)$. Depending on the options, the IoT device (agent) *l* selects the available node with resource availability, the computation resource and transmission power.

Then the agent obtains a reward that evaluates the performance of selected actions. We consider the computation cost $\mathcal{Z}_l(t)$ of agent l as the reward function $r_l(t)$, which is a negative reward. Particularly, the goal of the learning agent in DRL is to maximize the long-term reward $\vartheta_l(t)$ by selecting suitable actions at time slot *t*. Therefore the sum of discounted long-term reward is expressed as;

$$\vartheta_l(t) = \sum_{\iota=0}^{\infty} \lambda^{\iota} r_l(t+\iota+1),\ \forall l \in \mathcal{L}, \tag{32}$$

where ι is time slot index from time step *t*, $\lambda = [0,1]$ denotes the discount factor. Let us define the set of offloading decisions, computation capacity, and transmission power decision taken by agent *l*, as $\mathcal{A}_l(t) = \chi_{lx} \times \mathcal{F}_{lx} \times \mathcal{P}_{lx}$, the goal of each agent l is to obtain optimal action $a_l^*(t) = \{\alpha_{lx}^*, f_{lx}^*(t), p_{lx}^*(t)\} \in \mathcal{A}_l(t)$, which can maximize the long-term reward. The action of IoT device l at time slot t is $a_l(t) = (\chi_{lx}^m(t), f_{lx}(t), p_{lx}(t)) \in \mathcal{A}_l(t)$. The optimal action selection policy is expressed as;

$$a_l^*(t) = \underset{a_l \in \mathcal{A}_l}{\operatorname{argmin}}\, \vartheta_l(t) \tag{33}$$

To obtain a solution for the problem, we apply MADRL approach.

A. Stochastic Game-Based Problem Formulation

In this subsection, we model the formulated problem in (33) as a Markov game or stochastic game in a multi-agent scenario [32]. The Markov chain is a mathematical model that describes the dynamics of states in a stochastic game in which each player/agent has only one action in each state.

Definition 1: A finite state Markov chain is a discrete stochastic process, which can be described as follows: Let a finite set of states $S = \{s_1, s_2, \ldots, s_L\}$ and an $L \times L$ transition matrix $\mathbb{P}$ with each entry $0 \le \mathbb{P}_{i,j}$ and $\sum_{i=1}^{l} \mathbb{P}_{i,j} = 1$ for any $1 \le i \le l$. The process starts in one of the states and moves to another state successively. Assume that the chain is currently in state s_i. The probability of moving to the next state s_j is

$$Pr\big\{s(t+1) = s_j | s(t) = s_i\big\} = \mathbb{P}_{i,j} \tag{34}$$

Definition 2: A stochastic game can be defined as the following tuples: $\Theta = (\mathcal{S}, \mathcal{L}, \mathcal{A}, \mathbb{P}, \mathcal{R})$ where: $\mathcal{S}$ stands for the state with $\mathcal{S} = \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_L$, is the set of states for agent *l*, $\forall l \in \mathcal{L}$ and $\mathcal{L}$ is set of players (agents). $\mathcal{A}$ represents the joint action set and $\mathcal{A}_l$ is also the action set of player agent *l*. $\mathbb{P}$ denotes the state transition probability function which depends on the actions of all players. Particularly, $\mathbb{P}(s_l(t), a, s_l(t+1)) = Pr\{s_l(t+1)|s_l(t), a\}$ represents the probability of transitioning to the next state $s_l(t+1)$ from the state $s_l(t)$ by executing the joint action a with $a = \{a_1, a_2, \ldots, a_L\} \in \Phi$, and $\mathcal{R} = \{R_1, R_2, \ldots, R_L\}$, where $R_l = \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward value function of player *l*.

In the A2G network environment, each edge IoT device acts as an agent, and chooses an action from a set of actions and then receives the reward at time slot *t*. We describe the states, actions and rewards as follows.

a) State Space: The state $s(t) \in \mathcal{S}$ consists of task profile, UAV network status indicator (includes resource, coverage), SINR of both BS and UCH, power of agent *l*, and task type either delay-sensitive or delay-tolerance. Therefore, we define the state in our scenario as follows,

$$s_l(t) = \{\bar{\gamma}_{lx}(t), \mathcal{T}_l(t), \mathcal{P}_l(t), \nu_u(t), \mathcal{I}_l(t)\}, \tag{35}$$

where $\bar{\gamma}_{lx}(t) \in \{0,1\}$ indicates the SINR between the agent and computational node and is expressed as;

$$\bar{\gamma}_{lx}(t) = \begin{cases} 1, & \text{if } \big|\alpha_{lx}^m(t)\big|\gamma_{lx}(t) \ge \gamma' \quad \text{and} \quad s_{lx}^m(t) = 1 \\ 0, & \text{otherwise,} \end{cases}$$

where γ' is the QoS threshold. $\mathcal{T}_l(t)$ indicates the task information $d_l(t) \in \mathcal{R}_+$ input tasks size, $\mathcal{P}_l(t) \in \{0,1\}$ indicates the transmit power $p_l^{tr}(t)$ of agent *l* as;

$$\mathcal{P}_l(t) = \begin{cases} 0, & \text{if } P_{l,z}^T(t) = 0 \\ 1, & \text{otherwise} \end{cases}$$

where $\nu_u(t)$ indicates the UAV network condition in terms of resources and coverage and is expressed as;

$$\nu_u(t) = \begin{cases} 0, & \text{if } P_{u,z}^T(t) = 0 \quad c_{ku}(t) = 0, \psi_u(t) = 0 \\ 1, & \text{otherwise,} \end{cases}$$

where $c_{ku}(t)$ is network coverage of UCH *u* and depends on (29). $\mathcal{I}_l(t) \in \{0,1\}$ indicates the task is either time-sensitive or delay tolerant.

b) Action Space: Each IoT device or agent selects its action according to the current policies and other agents' experience information. The agent selects the computation resource, transmission power, and computational node.The action of agent *l* is defined as;

$$a_l(t) = \{\chi_{lx}(t), f_{lx}(t), p_{lx}(t)\}, \tag{36}$$

where $\chi_{lx}(t) \in \{-1, 0, 1\}$ indicates the offloading decision or association. $f_l(t) = \{1, 2, \ldots, F_x^{max}\}$ selects the computational resource from resource blocks of BS or UCH and $p_l(t) = \{1, 2, \ldots, P_x^{max}\}$ selects the transmission power resource from the resource blocks of BS *n* or UCH *u*, where $x \in \mathcal{X}, n, u \subset x$. The agent's action at time slot *t* is $a_l(t) \in \mathcal{A}_l = \chi_{lx} \times \mathcal{F}_{lx} \times \mathcal{P}_{lx}$. Therefore, the action space in the stochastic game is $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times, \ldots, \mathcal{A}_L$.

c) Reward: An agent can make its own decision in a decentralized execution manner to maximize the reward in the environment. To ensure the QoS in the A2G network, we design the reward function, which minimizes the overall computation costs of tasks in different computation nodes. At time slot *t*, the reward is the sum of the rewards of all agents. The reward of each agent consists of energy cost and time delay cost obtained after task completion at the time slot *t*, and is expressed as:

$$r_l(t) = \Big\{r_l^e(t), r_l^T(t)\Big\} \tag{37}$$

The energy cost of reward is expressed as:

$$r_l^e(t) = \begin{cases} E_l^{loc,exe}(t), & \text{if } \chi_{lx}(t) = 0, \mathcal{P}_l(t) = 0 \\ E_{lx}^{bs,uav}(t), & \text{if } \chi_{lx}(t) = 1 | -1, \bar{\gamma}_{lx}(t) = 1, \end{cases}$$

where $r_l^e(t) = (E_l^{loc,exe}(t) - E_{lx}^{bs,uav}(t))$. We take the difference between the local computation energy cost and currently obtained energy cost as part of the reward. When the agent processes computation tasks locally, it means the energy reward will be 0. When the tasks are offloaded to the local MEC server or UCH, a positive reward will be received if a lower energy cost is obtained than the local calculation. Otherwise, a negative reward will be received. The delay cost function of the reward is expressed as follows:

$$r_l^T(t) = \begin{cases} T_l^{loc}(t), & \text{if } \chi_{lx}(t) = 0, \mathcal{P}_l(t) = 0 \\ T_{lx}^{bs,uav}(t), & \text{if } \chi_{lx}(t) = 1|-1, \text{ and } \bar{\gamma}_{lx}(t) = 1, \\ & \text{execution delay and transmission} \end{cases}$$

Therefore, the difference between local time delay and the current time delay reward is time delay cost expressed as:

$$r_l^T(t) = T_l^{loc}(t) - T_{lx}^{bs,uav}(t) \tag{38}$$

Besides, we consider that agents cooperate with other agents. To enhance the convergence of the proposed MADDPG-based algorithm, we define a negative value for the reward. We formulate the reward by adding the weight index of energy and delay cost as:

$$r_l(t) = -\Big(\xi r_l^T(t) + \phi r_l^e(t)\Big), \tag{39}$$

where ϕ and ξ represent the weight index factor of energy cost difference reward and time difference reward of agent l, respectively. If the computation cost difference is negative, the reward will be positive , otherwise we obtain a negative reward value at time slot t. The overall reward of all agents at time slot t expressed as:

$$r(t) = \sum_{l=1}^{L} r_l(t), \forall l \in \mathcal{L} \tag{40}$$

The main objective is to minimize the computation cost of tasks in the A2G network. We maximize the long-term reward as:

$$\pi* = \arg\max_{a} \mathbb{E}_\pi \left[\sum_{t=1}^{+\infty} \lambda^{t-1} r(t) \right]. \tag{41}$$

A policy $\pi_l(s_l, a_l)$ associates with $s_l \in \mathcal{S}_l$ and $a_l \in \mathcal{A}_l$ to a probability $\pi_l(s_l, a_l = Pr(a(t) = a|s(t) = s) \in [0, 1]$. Particularly, the IoT device l at state s_l has mixed strategy expressed as $\pi_l(s_l) = \{\pi_l(s_l, a_l)|_{a_l \in \mathcal{A}_l}\}$. Therefore, in stochastic game model, the strategy of L players is defined in vector as $\pi = (\pi_l(s_l), \pi_2(s_2), \ldots, \pi_L(s_L))$. The main goal of IoT device is learn the optimal strategy $\pi*$ from a given state $s_l \in \mathcal{S}_L$. Let us define other players' device strategies as $\pi_{-l}^* = \{\pi_l^*, \ldots, \pi_{l-1}^*, \pi_{l+1}^*, \ldots, \pi_L^*\}$. Depending on probabilistic policies we formulate the optimization goal of each IoT device l in (32) into expected discounted $\mathbb{E}$ equation as:

$$\vartheta_l(t)(s_l, \pi) = \mathbb{E}\left[\sum_{\iota=0}^{\infty} \lambda^\iota r_l(t + \iota + 1)|s_l(t) = s_l, \pi \right]. \tag{42}$$

Definition 3: An NE is a group of L players' optimal strategies $\pi* = \{\pi_1^*, \pi_2^*, \ldots, \pi_l\}$. Each individual player's strategy is a best-response to other players. Therefore, $\forall l$, the strategy π_l^* is expressed as

$$\vartheta_l\big(s_l, \pi_l^*, \pi_{-l}^*\big) \leq \vartheta_l\big(s_l, \pi_l, \pi_{-l}^*\big), \tag{43}$$

where π_l is the set of all possible strategies taken by IoT device l, which consists of rewards policies such as $\pi_l = \{\pi^e, \pi^T\}$.

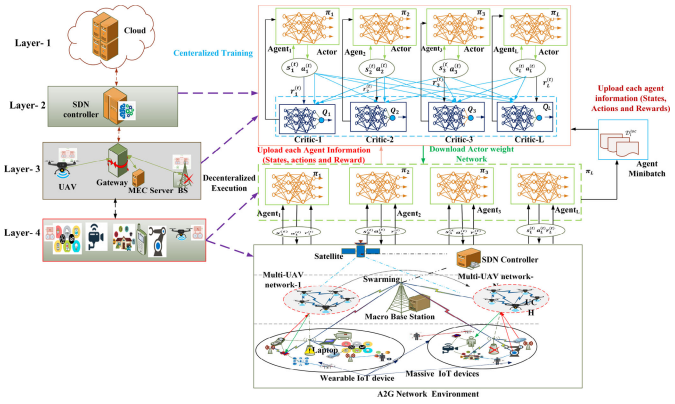


Fig. 2. MADDPG based Multi-UAV enabled IoT edge network framework.

V. MADDPG Based Solution

A. MADDPG Environment and Algorithm

Previously, the traditional single-agent DRL algorithms were useful for small scale problems. However, the A2G network environment has wide-ranging problem space, multi-agent environment, and unpredictable dynamic change at time slot t. Fig. 2 depicts the framework of MADDPG-based multi-UAV enabled IoT edge network using the current network technology such as SDN.

Layer-1 is an application layer on the cloud that has several cloud service applications and supports many application systems.

Layer-2 is a control layer that contains the SDN controller used to manage the data layer devices and edge computing servers in a centralized manner. This layer uses an OpenFlow protocol. The SDN controller also allocates resources such as spectrum, computation and transmission power based on different network requirements.

Layer-3 is a data layer which contains switches, BSs, gateways, and UCH, controlled by the SDN controller. The controller manipulates the resource allocation, computation offloading decisions, and other decisions.

Layer-4 has ultra-dense heterogeneous IoT devices, which consist of essential end-user smart devices. In our scenario, we consider each smart edge device as an agent in the learning process. This framework adopts centralized training with a decentralized execution model, and each IoT device l at the edge layer is an autonomous agent. In this sequel, we refer to IoT devices as agents and represent either entity as l. The agents have actor-network as the decentralized execution, and a critic network is the centralized training parts. The centralized training (critic function or action-value function) takes place mainly either at UCH or BS, and sometimes at the central controller at time slot t.

MADDPG-based algorithm is suitable for the multi-agent environment, as agents can learn cooperatively and improve network performance. The MADDPG algorithm is one of the MADRL schemes [32], which is a modified actor-critic network with the DQN method that can handle the dynamic environment and is useful for cooperative policy learning of multi-agents with continuous action space. The critic function also uses the action policies of other agents, and agents can make a decision independently using their strategies and observations. Policy parameters are updated by an individual. To ensure the QoS of the IoT devices and improve network performance, we deploy a multi-UAV network, which acts as ABS with flying MEC (FMEC) server to provide different services for IoT devices at the IoT edge network level. In this work, we focus on overall task computation cost minimization in terms of energy, time delay and considering UAV resource constraints in a dynamic A2G network environment.

The main idea of MADDPG is to learn a centralized critic function (action-value function) $Q_l^{\pi}(\mathcal{K}, a_1, ...a_L)$ for an l^{th} agent, where $\{a_1, \ldots, a_L\} \in \mathcal{A}$ are the representation of all agents actions and each agent has the global training weight information. The observation of agents is represented as $\mathcal{K} = \{\mathcal{O}_1, \ldots, \mathcal{O}_L\}$, the set of policies of all agents as $\pi = \{\pi_1, \ldots, \pi_L\}$, and policy with parameter of L agents is denoted by $\theta = \{\theta_1, \ldots, \theta_L\}$. In the MADDPG algorithm, the actor and critic network will be updated after each training episode. The actor-network can be updated using the gradient method calculated as follows:

$$\nabla_{\theta_l}(\theta_l) = \mathbb{E}_{\mathcal{K},a\sim\mathcal{D}}\big[\nabla_{al}\theta_l^{\mu}(\mathcal{K}, a_1, ., a_L) \nabla_{\theta} l\mu\theta_l(\mathcal{O}_l)|_a l = \mu\theta_l(\mathcal{O}_l)], \quad (44)$$

where $\mathcal{D}$ is the experience replay buffer which stores all agent experiences including $(\mathcal{K}, \mathcal{K}', a_1, \ldots, a_L, r_1, \ldots, r_L)$. The critic function θ_l^{μ} is updated as the following expression:

$$\mathcal{L}(\theta_l) = \mathbb{E}_{\mathcal{K},\mathcal{K}',a_1,\ldots,a_L,r_1,\ldots,r_L}\Big[\big(\theta_l^{\mu}(\mathcal{K}, a_1, \ldots, a_L) - y\big)^2\Big], \quad (45)$$

where $y = r_l + \gamma\theta_l^{\mu'}(\mathcal{K}', a_1', \ldots, a_L')|_{a_j'=\mu_j'(\mathcal{O}_j)}$. The set of target policies with delayed parameters θ_l' is denoted as $\mu' = \{\mu\theta_1', \ldots, \mu\theta_L'\}$.

The mode of operation of Algorithm 1 is presented as follows: Each agent downloads the actor-network training weights from the central trainer and loads on its actor-network. At the beginning of each time slot t, an agent l obtains observations or state s_l. The action is executed in a decentralized manner by each agent independently and obtains the current reward r_l and the transition state s'. Each agent has its own local experience replay buffer $(\mathcal{D}_l^{loc})$ storing tuples $\{s_l, a_l, r_l, s'\}$. The agent l can take action $a_l(t)$ according to the local observation state $s_l(t)$ from the environment and return reward $r_l(t)$. When the communication channel is available and the BS is not overloaded by the agent l, upload a local information $\{s_l(t), a_l(t), r_l(t)\}$ is uploaded onto the BS for successive training, else agent l uploads onto associated UCH u. Every agent will upload its experience relay buffer $\mathcal{D}_l^{loc}$ onto the selected computational node and download the global actor train weight network or experience relay buffer $\mathcal{D}^{uav}, \mathcal{D}^{bs}$ from the BS or UCH. Then, the agent l updates its policy and makes a decision to either offload the task through subchannel or compute locally. This depends on the current observation and policy at time slot t. A collection of tuples such as state, action, reward and transition state of all agents are stored on global experience replay buffer at each time slot t and denoted as $\mathcal{D} = \{e_1, e_2, \ldots, e_t\}$, where $e_t = \{s_t, a_t, r_t, s_{t+1}\}$, $\mathcal{D}^{uav}, \mathcal{D}^{bs} \in \mathcal{D}$. In the centralized training, the critic network uses inputting of action and state of all agents to obtain the approximate Q-value function of current action of agent l.

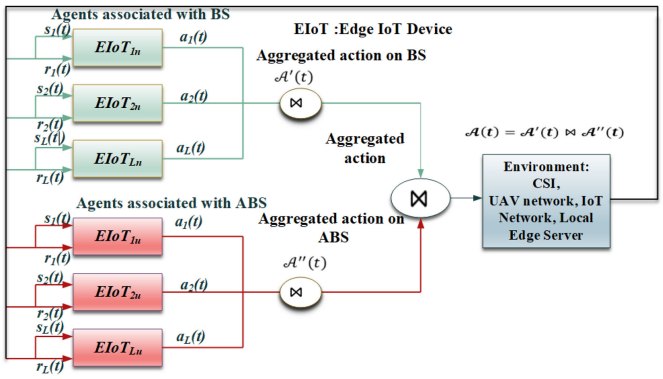


Fig. 3. MADRL framework in A2G Network.

B. MADRL Framework in A2G Network

Fig. 3 shows the MADRL framework of multi-UAV enabled IoT edge network in A2G network and depicts the main components of MADRL. Each agent l has its locally observed information or state $s_l(t)$ at each time slot t and takes the action $a_l(t) \in \pi_l(s_l(t))$. Then, the environment enters into new states. According to the current strategies and observed information of an agent,[4] it will select only one computational node in time slot t. However, the actions of all agents are aggregated after agents obtain the reward. Let the aggregated actions at BS be $\mathcal{A}'(t) = \{a_1(t) \bowtie a_2(t), \ldots, \bowtie a_l(t)\}$, and at ABS be $\mathcal{A}''(t) = \{a_1(t) \bowtie a_2(t), \ldots, \bowtie, a_l(t)\}$. Then, the final action set is joined as $\mathcal{A}(t) = \mathcal{A}'(t) \bowtie \mathcal{A}''(t)$ on the SDN controller at time slot t. Explicitly, a single player in the stochastic game is modeled as a Markov decision process (MDP). Moreover, the decision problem faced by a player in a stochastic game when all other players choose a fixed profile of stationary strategies is equivalent to an MDP [32].

The deep neural network (DNN) algorithm has two fully connected layers with activation function and has a number of output layers which hold the possible actions $(\chi_{lx}(t), f_{lx}(t), p_{lx}(t))$ at time slot t. The critic-network also has the same DNN hidden layer values and activation function but has one output layer. The inputs of the critic-network are the state s and action a of all agents, and generate the Q-value function $Q^{\pi}(s(t), a(t))$. To update these two networks, the critic network Q_l^{μ} is updated by the loss function in (45), and the actor-network is updated by minimizing the policy gradient

[4]Note that each agent are connected with one computational node in the time slot t and it's autonomous.

Algorithm 1 MADDPG Algorithm for Computation Offloading and Resource Allocation in A2G Network Environment

Initialize: Global replay buffer $\mathcal{D}$ at the controller layer and $\mathcal{D}^{uav}, \mathcal{D}^{bs}$ at the UCH and BS;
Initialize: The parameters of actor and critic network with random weights θ_l;
for episode $= 1$ to 2000 **do**
  All agents observe initial state $S = \{s_1, s_2, \ldots, s_L\}$
  for $t = 1$ to 200 **do**
    Each agent l selects a random action a_l based on the probability ε, else select action $a_l = \pi_{\theta_l}(s_l)$;
    All agents execute action $a(t) = \{a_1(t), a_2(t), \ldots, a_L(t)\}$ and observe rewards $r(t) = \{r_1(t), r_2(t), \ldots, r_L(t)\}$, and the new state $s_l(t+1) \sim s_l'$
    Store the tuples $\{s_l(t), a_l(t), r_l(t), s_l'(t+1)\}$ in $\mathcal{D}_l^{loc}$
    if Agent l is associated/offloaded ($\chi_{lx}(t) = -1$) onto UCH **then**
      Upload the tuple value from $\mathcal{D}_l^{loc}$ to UCH
      Merge $\mathcal{D}_l^{loc}$ into $\mathcal{D}_u^{uav}$ at UCH
    else
      Upload the tuple value from $\mathcal{D}_l^{loc}$ into BS n ($\chi_{lx}(t) = 1$)
      Merge $\mathcal{D}_l^{loc}$ into $\mathcal{D}_n^{bs}$ at BS n;
    end if
    Download either $\mathcal{D}_n^{bs}$ or $\mathcal{D}_u^{uav}$ from BS and UCH
    $s_l \leftarrow s_l'$;
    for agent $l = 1$ to L **do**
      Sample a random mini-batch of H samples tuples (s^j, a^j, r^j, s'^j) from either $\mathcal{D}_u^{uav}$ or $\mathcal{D}_n^{bs}$;
      Set $y^j = r_l^j + \gamma Q_l^{\pi'}\left(S'^j, a_1', \ldots, a_L'\right)|_{a_k' = \pi_k'(s_k^j)}$;
      Update the critic-network by minimizing the loss (45); $\mathcal{L}(\theta_l) = \frac{1}{H}\sum_j \left(y^j - Q_l^{\pi}(s^j, a_1^j, \ldots, a_L^j)\right)^2$;
      Update actor-network using the sample policy gradient (44);$\nabla_{\theta_l} J \approx \sum_j \nabla_{\theta_l}\mu_l(o_l^j)\nabla_{a_l} Q_l^{\mu}\left(\mathcal{K}^j, a_1^j, \ldots, a_L^j\right)|_{a_l = \mu_l(\mathcal{O}_l^j)}$
    end for
    Update the target network parameters for each agent l:
    $\theta_l' \leftarrow \tau\theta_l + (1-\tau)\theta_l'$;
  end for
end for

of agent l, where H is a random mini-batch size of samples and is expressed as:

$$\nabla_{\theta_l} J \approx \frac{1}{H}\sum_j \nabla_{\theta_l}\mu_l\left(o_l^j\right)\nabla_{a_l} Q_l^{\mu} \times \left(K^j, a_1^j, \ldots, a_l^j\right)|_{a_l = \mu_l\left(o_l^j\right)}, \quad (46)$$

where j is the index of samples. The MADDPG algorithm, as presented in **Algorithm 1** has two procedures: collecting observed data and training procedures. We first initialize the replay buffer, actor-network, critic network parameters with weight (line 1-2), and defined number of episodes and training time steps. Then, the agents collect observed data (line 3-18). Each agent executes the action, obtains the reward, and generates a new state (line 6-7). The experience is stored in the experience replay buffer. In the training step (line 18-25), we use a policy training which uses batch sampling from the replay buffer. Then the actor-network and the critic-network are updated based on a randomly selected sample.

C. Algorithm Analysis

i) Convergence: In our proposed algorithm, we adopt the gradient method to update weight θ_l for actor-network and critic-network, while the learning rate decreases with iterations. Then, after a finite number of iterations, the weight of θ_l will converge to a particular value, which ensures the convergence of our proposed algorithm. The theoretical convergence analysis of DRL learning (NN) is challenging before training. Therefore, the convergence of our algorithm can be observed by simulations.

ii) Computational Complexity Analysis: The DNN of the actor can be represented in matrix multiplication. Let us define N and $|L|$ as a number of hidden layers and dimensions of the output. Then, the complexity of each actor is estimated as $\mathcal{O}(|L|^2 N)$. The computational complexity between agents is $\mathcal{O}(|L|)$, and the final issuing policy at each time step is expressed as $\mathcal{O}(|L|^2 N)$. In our proposed framework, increasing the number of the agents should not influence each agent's computational complexity. Although, when the agent cost Z_l is increasing, it will increase the training time estimated as $\mathcal{O}(Z_l|L|N)$, $\mathcal{O}(Z_{eps}HZ_l N|L|^2)$ for the complexity of critic networks and training procedure, respectively. The training algorithms is affected by the number of agents L, the number of training episodes Z_{eps} and batch size H.

VI. Performance Evaluation

In this section, we demonstrate the effectiveness of the proposed MADRL based computation offloading, and resource allocation for multi-UAV enabled IoT edge network via simulations.

A. Scenario Configuration

The deployment and parameters configuration of multi-UAV network and IoT network commonly depend on the works in [22], [29], [31], [48]–[50]. We consider the multi-UAV network in cluster formation with the UAVs connected in D2D communication mode. The UAV cluster is deployed in a small cell area with radius $r_u = 800m$, where the ground IoT devices are randomly and uniformly distributed in small cells and UAVs are assumed to fly at a fixed altitude $H_u = 100m$ [51]. The subchannel bandwidth is $\frac{W}{M} = 80$ KHz, and the QoS threshold (minimum SINR for agent) $\bar{\gamma} = 3.5dB$. For probabilistic model, $\iota = 9.61, \varrho = 0.16$, carrier frequency is $f = 2$ GHz, path-loss $\eta^{LoS} = 1$, and $\eta^{NLoS} = 20$. The pathloss model for M2M and D2D links are calculated as $128.1 + 36.6log_{10}(D_{l\bar{l}})$ and $148.1 + 40log_{10}(D_{l\bar{l}})$, respectively [52].

TABLE III
SIMULATION PARAMETERS

Parameter	Value
Number of UCHs (C)	2
Number of BSs (N)	2
Number of cells (K)	2
Number of channels (M)	10
Maximum power level (Z)	3
Max. transmission power of UCH (P_u)	125dBm
Noise power (δ)	-80dBm
Computation capacity of UCH (F_u)	15GHz/sec
Computation capacity BS (F_n)	20GHz/sec
Maximum transmission power of IoT device (P_l)	10dBm
Distance between IoT Devices ($D_{l\bar{l}}$)	50m
Required CPU (c_l) to compute input task	[0, 1.5]GHz
Learning rate of actor-network, critic-network	$1e^{-4}$ $3e^{-4}$
Discount factor λ	0.99
Soft update factor τ	$1e^{-3}$

We implement the simulations using Python 3.6 environment with Tensorflow 2.0 on a computer with a core i7 CPU running on a processor speed of 2.4GHz and 16GB RAM. We consider a 500m $\times$ 500m BS coverage in different small cells to provide wireless services. There exists two small cells with a radius of 500m between them. The number of IoT devices L is set as 30 in one small cell and 20 in another. The input task size of each IoT device d_l is randomly distributed from 50Kb to 20Mb.

We deploy a fully connected NN with critic-network and actor-network in this proposed multi-agent approach. For each agent, we deploy two hidden layers in both the actor and the critic NNs set as 256 in the first hidden layer and 128 in the second layer. We set the mini-batch size as 256, and a replay buffer size as 10^7. We set agent action selection probability $\epsilon \in (0, 1)$ weight index of time delay $\xi = 0.65$, and energy weight index $\phi = 0.35$. We utilize ReLU as the activation function for the hidden layers and sigmoid function is employed at the output layer. To optimize the loss function, we use the AdamOptimizer. The learning framework is constructed from a group of IoT devices, UCHs, and BSs. The agents execute the trained data/test to obtain the desired output. Each agent executes its own observed information using the trained model. Other simulation parameters are summarized in Table III.

To evaluate the performance of our proposed framework, we consider the following metrics:

a) Average computation cost: This is the mean computation costs of all agents' computation cost in the system.

b) Task offloading rate and drop rate: This indicates the percentage of offloading task and the failed computing tasks that are generated from agents. These two metrics are used to determine the resource allocation and utilization in the system.

c) Task execution time delay and energy consumption: These play a vital role in network performance and improve the QoS. Both metrics indicate from transmission up to the completion of offloading tasks.

To compare the performance of our proposed algorithm in multi-UAV enabled IoT edge network, we provide four other baselines: DDPG [53], Dueling-DQN [22], DQN [22], and greedy algorithms [54]. However, DQN cannot directly support the large problems with continuous action space. Therefore, it discretizes the continuous action space to solve our problem.

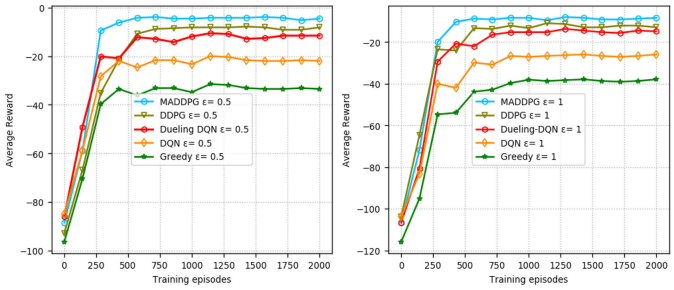


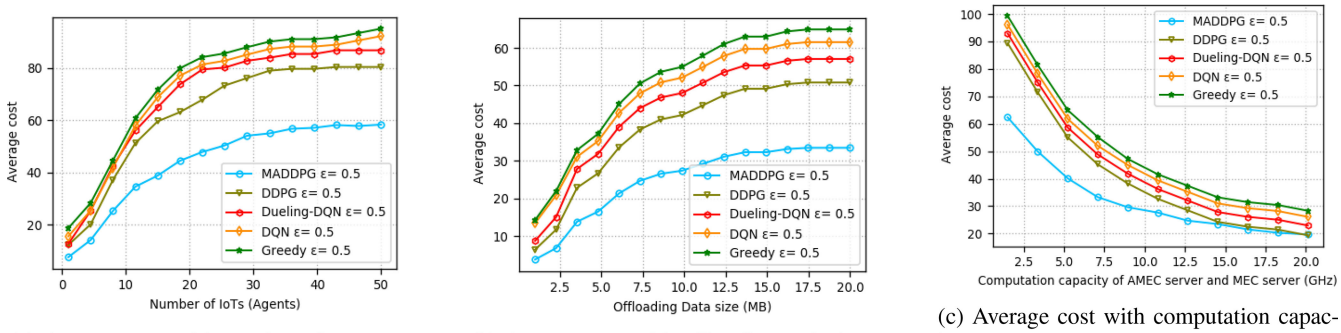
Fig. 4. Average system reward with episodes.

B. Convergence Analysis

Fig. 4, shows the convergence on average reward with episodes, where the action selection probability $\epsilon = \{0.5, 1\}$. When the value is $\epsilon = 1$, it indicates different agents access only a ground BS with local MEC server and $\epsilon = 0.5$ indicates the agents have an opportunistic computational node ABS or BS. The average reward values of MADDPG, DDPG, Dueling-DQN, DQN and greedy increase continuously with increasing number of episodes until convergence. Moreover, MADDPG algorithm becomes stable at episode 250 with $\epsilon = 0.5$ and close to 400 with $\epsilon = 1$. The other four algorithms become stable after 500 of training episodes with both $\epsilon = 0.5$ and $\epsilon = 1$. The average reward value of MADDPG is the highest and its convergence is the fastest. Therefore, we conclude that the MADDPG outperforms the other baselines in both action selection probability values ($\epsilon = \{0.5, 1\}$) and maximize the long-term rewards (minimize cost). The MADDPG algorithm can learn the policy of cooperation and reduce the UAVs' resource wastage rather than keeping their own QoS and maximizing the global reward.

C. Average Cost

We discuss the average cost in terms of several number of IoT devices, the computation capacity, and the size of offloading tasks with an action selection probability (ϵ) value of 0.5. Fig. 5(a) shows the performance of the proposed algorithm in the A2G network to minimize computation costs. We consider the ABS and ground BS are available and take the mean cost of both. We observe the impact of agents on minimizing the computation costs and ensuring agents' QoS in multi-agent systems. From Fig. 5a, we observe that as the number of agents (IoT devices) increases, each agent's average cost increases gradually. This implies that the costs also increase with several agents. Aside the number of agents increasing, there are different possibilities, such as massive traffics. Due to this, there are high transmission power and time delay cost. When we compare the percentage of each algorithm according to the maximum number of agents, the MADDPG algorithm reduces the average cost by 31.7488%, 39.188%, 44.993%, 55.621% for DDPG, Dueling-DQN, DQN and greedy, respectively. Therefore, we conclude that when the number of agents



(a) Average cost with number of agents.

(b) Average cost with offloading task size.

(c) Average cost with computation capacity.

Fig. 5. Comparison of average cost with number of agents, size of offloading task and computational capacity.

increases, the MADDPG algorithm outperforms the four algorithms in terms of the number of agents learned in cooperative policy through the MADDPG algorithm.

Fig. 5(b) depicts the average cost increases with increasing offloading data size. It implies that the computation cost in the system also increases due to a massive amount of data offloaded from each agent. When we compare the percentage of each algorithm according to the maximum offloaded task size, the MADDPG algorithm reduces the average cost by 35.762%, 42.28%, and 55.4145%, 60.428% for DDPG, Dueling-DQN, DQN and greedy algorithms respectively. Therefore, the MADDPG algorithm can obtain the best reward to minimize costs than the other four algorithms, and also the computation costs are lower than the others.

In Fig. 5(c), we plot the average cost against the required number of CPU cycles per task of the four algorithms. It shows the impact of the CPU cycles in a learning environment that satisfies the agent's interests while keeping the minimum computation costs. In the beginning, the number of CPU cycles is less than 2 GHz. The agents suffered to offload tasks due to the high computation cost required and also time delay is higher than the energy cost. Besides, as the number of CPU cycles required increases, the agents obtain sufficient computation resources. Due to this, the agents increase their interest to offload tasks onto the available computation nodes to minimize the computation costs and improve QoS. Therefore, the MADDPG algorithm outperforms the four baseline algorithms in terms of obtaining the best long-term rewards.

D. Task Execution Delay and Energy Consumption

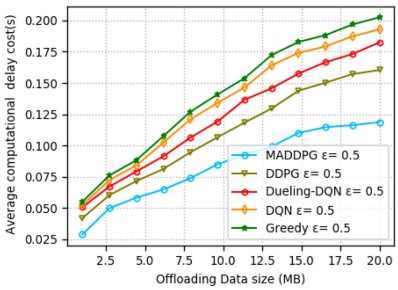
In this simulation, we discuss two essential metrics in the network system. Under the MADDPG algorithm, agents can choose the optimal computational node, avoiding traffic congestion and unnecessary costs. To demonstrate this, Fig. 6(a) shows the MADDPG algorithm has the lowest average time delay compared with DDPG, Dueling-DQN, DQN, and greedy algorithms. Besides, the time delay difference between the MADDPG algorithm and the other four algorithms grows regularly, which implies that the MADDPG algorithm outperforms the others under our scenario with increasing offloading data size. When we compare the percentage of each algorithm according to the maximum offloaded task size, the MADDPG algorithm reduces the time delay by 36.578%, 48.910%, 52.452% 62.367% for DDPG, Dueling-DQN, DQN and greedy, respectively. Therefore, when the offloading data size increases, the time delay increases in parallel.

We observe in Fig. 6(b), the impact of the offloaded data size in energy consumption at the IoT edge network. The energy cost of all algorithms increase with increasing size of offloaded data. We can observe from the figure that the MADDPG algorithm has better performance than the four baseline algorithms. Even if the computation task is to increase, the energy cost also grows slowly over the MADDPG algorithm than other algorithms. When we compare the percentage of each algorithm according to the maximum offloaded task size, the MADDPG algorithm reduces the energy consumption by 37.803%, 45.633%, 58.448%, 67.263% for DDPG, Dueling-DQN, DQN, greedy, respectively. Therefore, the cooperative computation offloading and resource allocation by the multi-agent algorithm outperforms single-agent DRL schemes and greedy scheme in massive IoT edge networks.

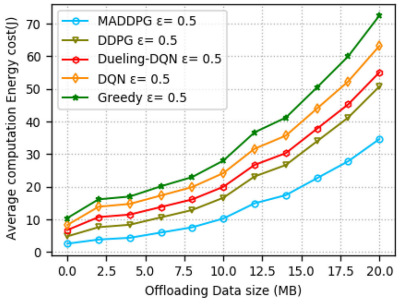
In Fig. 6(c) and Fig. 6(d), we show the impact of computation capacity in computation time and energy consumption on network performance. When the computation node's computing capacity increases, the computation time decreases, and the energy consumption also increases. This implies that the computation node power consumption increases due to many agents' decision to offload tasks onto ABS or BS at time slot t. However, the greedy, Dueling-DQN and DQN have less performance than DDPG algorithms in both time and energy costs, but the MADDPG algorithm performs better than DDPG. The cooperation amongst agents through the MADDPG algorithm reduces the time delay by 46.724%, 35.314%, 25.86%, and 19.44%, compared with greedy, DQN, Dueling-DQN, and DDPG, respectively. The energy consumption also reduces by 62.467%, 51.90%, 44.29%, and 32.44% for greedy, DQN, Dueling-DQN, and DDPG, respectively. Therefore, the multi-agent approach is better to reduce the computation time, and energy consumption than a single agent approach and heuristic approach in a multi-UAV enabled IoT edge network.

E. Offloading Rate and Task Drop Rate

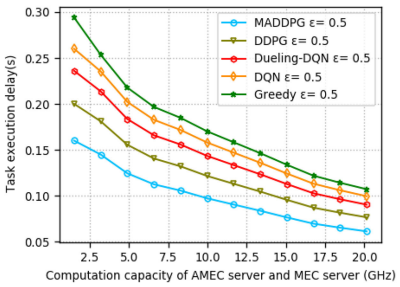
Fig. 6(e) shows the agents' task offloading rate decreases as the number of agents increases. We also compare the offloading rate in terms of the number of agents, when the action



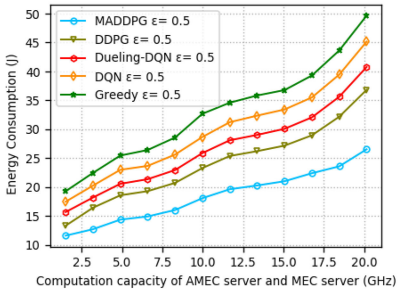
(a) Average time delay with offloading task size.



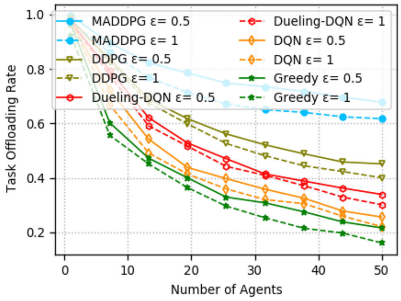
(b) Average energy consumption with offloading task size.



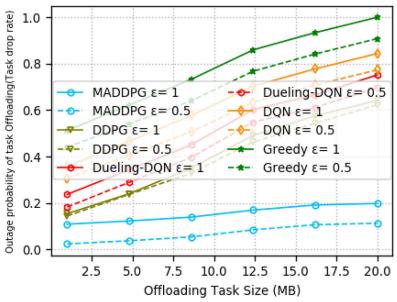
(c) Time delay cost with computation capacity.



(d) Energy consumption cost with computation capacity.

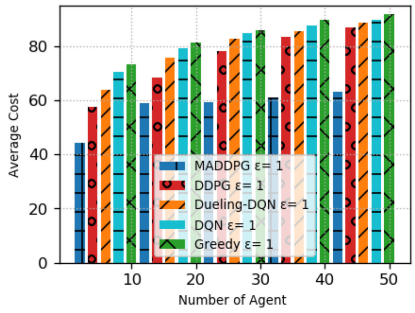


(e) Task offloading rate with the number of agents.

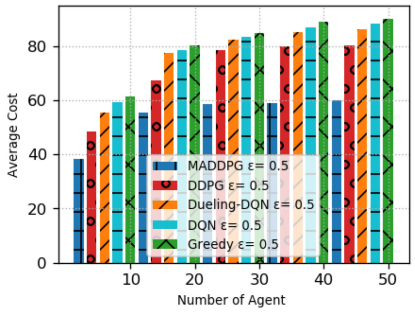


(f) Outage probability of task offloading with size of offloading task.

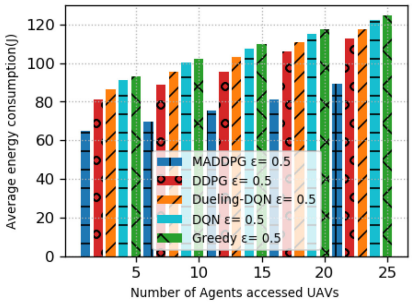
Fig. 6. Comparison of time delay, energy, offloading rate and task drop rate with number of agents, size of offloading task and computational capacity.



(a) Average cost without ABS ($\epsilon = 1$).



(b) Average cost with ABS ($\epsilon = 0.5$).



(c) UAVs energy consumption with agents.

Fig. 7. Comparison of average cost with agent ($\epsilon = 1$ and $\epsilon = 0.5$), UAV energy consumption.

selection probability is $\epsilon = 1$ and $\epsilon = 0.5$. We observe that different agents choose one of the two options ($\epsilon = 1$); the task offloading rate decreases due to the high computation costs to offload tasks. We also observe that the task offloading rate is decreased in the different results of all four algorithms in both options because some agents are far from the BS. This is because, BS traffic exists, and an emergency may have occurred. It consumes high transmission power and does not support the delay-sensitive IoT devices. However, when different agents are opportunistic ($\epsilon = 0.5$), i.e., choose ABS or BS at time slot t, the task offloading rate is higher than that of non-opportunistic ($\epsilon = 1$), i.e., BS only. In both options, the MADDPG algorithm outperforms DDPG by 40%, Dueling-DQN by 66.65%, DQN by 81.36% and greedy by 98.489% in terms of offloading rate when the agent chooses either ABS or BS ($\epsilon = 0.5$), and DDPG by 42.67%, Dueling-DQN by 68.845%, DQN by 85.41% and greedy by 97.173% when the agent action selection probability is $\epsilon = 1$, i.e., chooses BS only. Based on the above analysis, we conclude that using the MADDPG algorithm, the IoT edge network increases performance, minimizes computation costs, and improves the QoS of agents. Fig. 6(f) shows the relationship between outage task offloading (task drop rate) and the size of offloading tasks. The outage probability in our scenario indicates the availability of communication links between agents and computation nodes, computation node capacity, and network congestion. We observe that the outage probability of offloading tasks (task drop rate) increases as the size of offloading data increases. However, the MADDPG algorithm has the lowest outage probability of task offloading compared with the four algorithms, even if the agents' action selection probability is $\epsilon = 1$ or $\epsilon = 0.5$. However, the MADDPG algorithm slightly increases and has the lowest task drop rate when $\epsilon = 0.5$ or $\epsilon = 1$. Therefore, the MADRL approach is more applicable for multi-UAVs enabled IoT edge network to reduce task drop rates, minimize the computation cost and improve QoS, as well

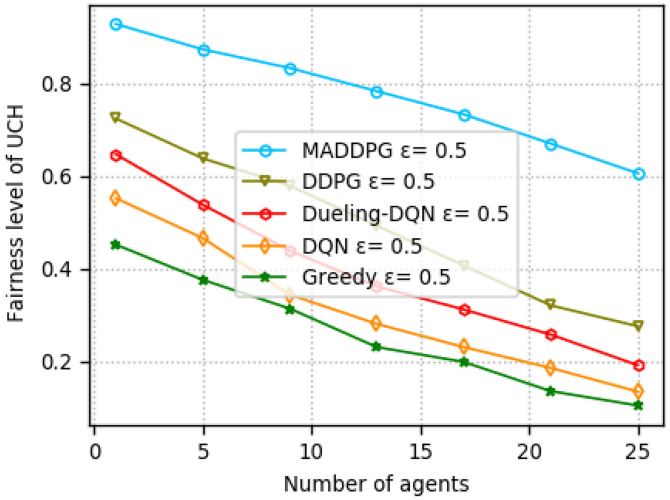


Fig. 8. UCH fairness level with number of agents.

as to reduce task drop rates due to different reasons such as high network traffic, low network coverage, and agent malfunctioning.

F. Analysis on Sample Cost With ABS and Without ABS

Fig. 7 shows the computation costs when action selection is $\epsilon = 1$ and $\epsilon = 0.5$. Fig. 7(a) shows that the agents select only the BS without ABS deployed ($\epsilon = 1$). In this figure, the average computation cost of the MADDPG algorithm is the lowest of the four algorithms. Fig. 7(b) shows the average computation cost in the A2G network environment, where the action selection $\epsilon = 0.5$ is reduced. In both options, the MADDPG algorithm outperforms the four baseline algorithms in minimizing the computation costs and improving the QoS of IoT devices in the IoT edge network. Fig. 7(c) shows the total energy consumption of UAV clusters to serve small number of agents/IoT devices in the IoT edge network. We observe that some of the agents chosen by the UAVs are far from the network, and the IoT edge network traffic is overloaded. The agents cooperate in a multi-agent approach that uses the MADDPG algorithm, and better utilize the resource of UAVs, and also consume low power than the other four algorithms. Figure 8 shows the impact of the number of agents/IoT devices on the UCH coverage level (UCH fairness level). We observe that the fairness level of UCH decreases rapidly in the four baseline schemes. The number of agents increasing would cause more difficulty in A2G communication services due to offloading tasks and consuming more UCH resources. However, the MADDPG scheme would cooperate with the agents to minimize the UCH burden and increase communication service. Hence, the fairness level of UCH declines slowly when the number of agents increases.

VII. Conclusion

In this paper, we studied the computation offloading and resource allocation problem in a multi-UAV enabled IoT edge network to minimize the computation costs. We deployed multi-UAVs when there is massive traffic (network congestion), wearable IoT devices are far from network coverage, and malfunctioning caused by disasters at the IoT edge network exists. We formulated the problem as a stochastic game for dynamic computation offloading and efficient resource allocation. To solve the formulated problem, we proposed a multi-agent DRL framework called the MADDPG algorithm that adopts a centralized training with a decentralized execution approach. Each IoT device acts as an agent to find its optimal policy and independently conducts a decision based on the MADDPG algorithm. To achieve these goals, the agents cooperate with each other and share experiences. Simulation results reveal that the proposed MADDPG algorithm ensures a better convergence than traditional single-agent algorithms and heuristic algorithms. Compared with single-agent and greedy methods, the average costs in terms of energy consumption and computation delay in our proposed method are lower by 38.643% and 55.621%, respectively. Moreover, the agent reward of MADDPG increased by 58.289% and 85.289% compared with single-agent and heuristic methods, respectively. Therefore, the proposed multi-agent DRL approach can efficiently allocate resources and dynamically obtain optimal offloading policies of agents cooperatively. Also, multi-UAVs with the multi-agent approach can also effectively guarantee the QoS for heterogeneous IoT devices and increase resource utilization at edge networks mainly for wearable IoT devices, ground network traffic, failure of BS, and emergencies. Future work will include blockchain technology and other related methods in the A2G network infrastructure.

References

[1] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of Things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios," *IEEE Access*, vol. 8, pp. 23022–23040, 2020.

[2] L. Chettri and R. Bera, "A comprehensive survey on Internet of Things (IoT) toward 5G wireless systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 16–32, Jan. 2020.

[3] X. Xu *et al.*, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 522–533, Jun. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18319770

[4] S. S. D. Ali, H. P. Zhao, and H. Kim, "Mobile edge computing: A promising paradigm for future communication systems," in *Proc. IEEE Region 10 Conf.*, 2018, pp. 1183–1187.

[5] G. Geraci, A. Garcia-Rodriguez, L. Galati Giordano, D. López-Pérez, and E. Björnson, "Understanding UAV cellular communications: From existing networks to massive MIMO," *IEEE Access*, vol. 6, pp. 67853–67865, 2018.

[6] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based Internet of Things services: Comprehensive survey and future perspectives," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 899–922, Dec. 2016.

[7] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-based IoT platform: A crowd surveillance use case," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 128–134, Feb. 2017.

[8] M. Gharibi, R. Boutaba, and S. L. Waslander, "Internet of drones," *IEEE Access*, vol. 4, pp. 1148–1162, 2016.

[9] X. Cao, J. Xu, and R. Zhang, "Mobile edge computing for cellular-connected UAV: Computation offloading and trajectory optimization," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jun. 2018, pp. 1–5.

[10] S. Zhu, L. Gui, J. Chen, Q. Zhang, and N. Zhang, "Cooperative computation offloading for UAVs: A joint radio and computing resource allocation approach," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 74–79.

[11] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[12] F. Al-Turjman, M. Abujubbeh, A. Malekloo, and L. Mostarda, "UAVs assessment in software-defined IoT networks: An overview," *Comput. Commun.*, vol. 150, pp. 519–536, Jan. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366419307637

[13] Z. Yuan, X. Huang, L. Sun, and J. Jin, "Software defined mobile sensor network for micro UAV swarm," in *Proc. IEEE Int. Conf. Control Robot. Eng. (ICCRE)*, 2016, pp. 1–4.

[14] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2018, pp. 1–6.

[15] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[16] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, 2018, pp. 1–6.

[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, Feb. 2019.

[18] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.

[19] L. Wang *et al.*, "Rl-based user association and resource allocation for multi-UAV enabled MEC," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 741–746.

[20] J. Li, Q. Liu, P. Wu, F. Shu, and S. Jin, "Task offloading for UAV-based mobile edge computing via deep reinforcement learning," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, 2018, pp. 798–802.

[21] R. Wang, Y. Cao, A. Noor, T. A. Alamoudi, and R. Nour, "Agent-enabled task offloading in UAV-aided mobile edge computing," *Comput. Commun.*, vol. 149, pp. 324–331, Jan. 2020.

[22] A. M. Seid, G. O. Boateng, S. Anokye, T. Kwantwi, G. Sun, and G. Liu, "Collaborative computation offloading and resource allocation in multi-UAV assisted IoT networks: A deep reinforcement learning approach," *IEEE Internet Things J.*, early access, Mar. 2, 2021, doi: 10.1109/JIOT.2021.3063188.

[23] S. Guolin, M. Seid, and S. Anokye, "Machine learning based unmanned aerial vehicle enabled fog-radio aerial vehicle enabled fog-radio access network and edge computing," *ZTE Commun.*, vol. 17, no. 4, p. 33, 2019.

[24] G. Peng, Y. Xia, X. Zhang, and L. Bai, "UAV-aided networks for emergency communications in areas with unevenly distributed users," in *Proc. IEEE Int. Conf. Commun. Syst. (ICCS)*, 2018, pp. 25–29.

[25] M. S. Munir, S. F. Abedin, N. H. Tran, Z. Han, E.-N. Huh, and C. S. Hong, "Risk-aware energy scheduling for edge computing with microgrid: A multi-agent deep reinforcement learning approach," *IEEE Trans. Netw. Service Manag.*, early access, Jan. 5, 2021, doi: 10.1109/TNSM.2021.3049381.

[26] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multi-agent deep reinforcement learning for joint multi-channel 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] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li, and L. Wang, "Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 146264–146272, 2019.

[28] J.Cui, Y.Liu, and A.Nallanathan, "The application of multi-agent reinforcement learning in UAV networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2019, pp. 1–6.

[29] 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.

[30] X. Liu *et al.*, "Transceiver design and multihop D2D for UAV IoT coverage in disasters," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1803–1815, Apr. 2019.

[31] N. Zhao *et al.*, "UAV-assisted emergency networks in disasters," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 45–51, Feb. 2019.

[32] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.

[33] A. Kaur and K. Kumar, "Energy-efficient resource allocation in cognitive radio networks under cooperative multi-agent model-free reinforcement learning schemes," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1337–1348, Sep. 2020.

[34] 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.

[35] S. T. Arzo, R. Bassoli, F. Granelli, and F. H. P. Fitzek, "Multi-agent based autonomic network management architecture," *IEEE Trans. Netw. Service Manag.*, early access, Feb. 16, 2021, doi: 10.1109/TNSM.2021.3059752.

[36] M. S. Munir, N. H. Tran, W. Saad, and C. S. Hong, "Multi-agent meta-reinforcement learning for self-powered and sustainable edge computing systems," *IEEE Trans. Netw. Service Manag.*, early access, Feb. 9, 2021, doi: 10.1109/TNSM.2021.3057960.

[37] M. Yan, G. Feng, and S. Qin, "Multi-rat access based on multi-agent reinforcement learning," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2017, pp. 1–6.

[38] M. Yan, G. Feng, J. Zhou, and S. Qin, "Smart multi-RAT access based on multiagent reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4539–4551, May 2018.

[39] T. Yuan, W. D. R. Neto, C. E. Rothenberg, K. Obraczka, C. Barakat, and T. Turletti, "Dynamic controller assignment in software defined Internet of Vehicles through multi-agent deep reinforcement learning," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 585–596, Mar. 2021.

[40] I. Althamary, C. Huang, and P. Lin, "A survey on multi-agent reinforcement learning methods for vehicular networks," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, 2019, pp. 1154–1159.

[41] X. Liu, J. Yu, Z. Feng, and Y. Gao, "Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing," *China Commun.*, vol. 17, no. 9, pp. 220–236, 2020.

[42] A. Shamsoshoara, M. Khaledi, F. Afghah, A. Razi, and J. Ashdown, "Distributed cooperative spectrum sharing in UAV networks using multi-agent reinforcement learning," in *Porc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, 2019, pp. 1–6.

[43] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: https://doi.org/10.1145/1355734.1355746

[44] S. Chandrasekharan *et al.*, "Designing and implementing future aerial communication networks," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 26–34, May 2016.

[45] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.

[46] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[47] X. Wang *et al.*, "Dynamic resource scheduling in mobile edge cloud with cloud radio access network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2429–2445, Nov. 2018.

[48] Y. Zhang, W. Xia, F. Yan, H. Cheng, and L. Shen, "Multi-agent reinforcement learning for joint wireless and computational resource allocation in mobile edge computing system," in *Proc. Int. Conf. Ad Hoc Netw.*, 2019, pp. 149–161.

[49] E. Yanmaz, R. Kuschnig, and C. Bettstetter, "Achieving air-ground communications in 802.11 networks with three-dimensional aerial mobility," in *Proc. IEEE INFOCOM*, 2013, pp. 120–124.

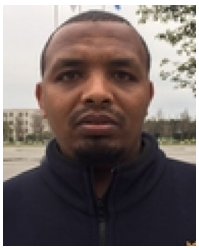
[50] E. W. Frew and T. X. Brown, "Airborne communication networks for small unmanned aircraft systems," *Proc. IEEE*, vol. 96, no. 12, pp. 2008–2027, Dec. 2008.

[51] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 2109–2121, Mar. 2018.

[52] S. Hakola, T. Chen, J. Lehtomäki, and T. Koskela, "Device-to-device (D2D) communication in cellular network—Performance analysis of optimum and practical communication mode selection," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2010, pp. 1–6.

[53] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: arXiv:1509.02971.

[54] F. Wei, S. Chen, and W. Zou, "A greedy algorithm for task offloading in mobile edge computing system," *China Commun.*, vol. 15, no. 11, pp. 149–157, Nov. 2018.



Abegaz Mohammed Seid received the B.Sc. degree in computer science from Ambo University, Ethiopia, in 2010, and the M.Sc. degree in computer science from Addis Ababa University, Ethiopia, in 2015, and the Ph.D. degree in computer science and technology from the University of Electronic Science and Technology of China in 2021. From 2010 to 2016, he worked with Dilla University, Ethiopia, as a graduate Assistant and a Lecturer, and worked with the College of Engineering and Technology as a Member of the Academic Committee and an Associate Registrar. He has published seven scientific conferences and journal papers. His research interests include a wireless network, mobile edge computing, blockchain, machine learning, vehicular network, IoT, fog computing, UAV network, IoT, and 5G/6G wireless network.

Gordon Owusu Boateng received the bachelor's degree in telecommunications engineering from the Kwame Nkrumah University of Science and Technology, Kumasi-Ghana, West Africa, in 2014, and the master's degree in computer science from the University of Electronic Science and Technology of China, where he is currently pursuing the Ph.D. degree. From 2014 to 2016, he worked under subcontracts for Ericsson, Ghana, and TIGO, Ghana. He is also a Member of the Mobile Cloud-Net Research Team, UESTC. His interests include mobile/cloud computing, 5G wireless networks, data mining, D2D communications, blockchain, game theory, and SDN.

Bruce Mareri received the B.Sc. degree in software engineering from Kenyatta University, Nairobi, Kenya, and the master's degree from the University of Electronic Science and Technology of China, Chengdu, China, where he is currently pursuing the Ph.D. degree in computer science. He got his Certification in Integrated Mobile Telecommunications Technology from the Emobilis Mobile Academy, Nairobi, in 2010. His interests include software defined networks, future networks, and ultra low latency 5G networks.

Guolin Sun (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in communication and information systems from the University of Electronic Science and Technology of China, Chengdu, China, in 2000, 2003, and 2005, respectively. After Ph.D. graduation in 2005, he has got eight years industrial work experience on wireless research and development for LTE, Wi-Fi, Internet of Things (ZIGBEE and RFID), cognitive radio, localization and navigation. Before, he joined the School of Computer Science and Engineering, University of Electronic Science and Technology of China as an Associate Professor in August 2012, he worked with Huawei Technologies, Sweden. He has filed over 40 patents, and published over 40 scientific conference and journal papers, acts as a TPC member of conferences. His general research interests include software defined networks, network function virtualization, and radio resource management. He currently serves as a Vice-Chair of the 5G oriented cognitive radio SIG of the IEEE Technical Committee on Cognitive Networks of the IEEE Communication Society.

Wei Jiang (Senior Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT) in 2008. In March 2008, he worked with the Central Research Institute, Huawei Technologies, for four years, in the field of wireless communications and 3GPP standardization. In September 2012, he joined the Institute of Digital Signal Processing, University of Duisburg-Essen, Germany, where he was a Postdoctoral Researcher and worked for EU FP7 ABSOLUTE project and H2020 5G-PPP COHERENT project. Since October 2015, he joined the Intelligent Networking Group, German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany, as a Senior Researcher and works for H2020 5G-PPP SELFNET project. He also works for the Department of Electrical and Information Technology, Technische University, Kaiserslautern, as a Senior Lecturer. He has authored more than 30 papers in top international journals and conference proceedings, and has 27 patent applications in wireless communications, most of which have already been authorized in China, Europe, USA, and Japan. He wrote a chapter "From OFDM to FBMC: Principles and Comparisons" for the book *Signal Processing for 5G: Algorithms and Implementations* (Wiley, 2016). His present research interests are in digital signal processing, multiantenna technology, cooperative communications, 5G, and machine learning. He served as a Vice Chair of IEEE TCCN Special Interest Group (SIG) "Cognitive Radio in 5G."