
Article:
Du, J., Kong, Z., Sun, A. et al. (4 more authors) (2023) MADDPG-based joint service placement and task offloading in MEC empowered air-ground integrated networks. IEEE Internet of Things Journal, 11 (6). pp. 10600-10615. ISSN 2327-4662

https://doi.org/10.1109/JIOT.2023.3326820

MADDPG-Based Joint Service Placement and Task Offloading in MEC Empowered Air-Ground Integrated Networks

Jianbo Du, Ziwen Kong, Aijing Sun, Jiawen Kang, Dusit Niyato, Xiaoli Chu and F. Richard Yu

***Abstract*—Multi-access Edge Computing (MEC) empowered Air-Ground Integrated Networks (AGINs) hold great promise in delivering accessible computing services for users and Internet of Things (IoT) applications, such as forest fire monitoring, emergency rescue operations, etc. In this paper, we present a comprehensive air-ground integrated MEC framework, where edge servers carried by Unmanned Aerial Vehicles (UAVs) will provide efficient computation services to IoT devices and User Equipment (which are collectively referred to as UEs). We aim to minimize the long-term average weighted sum of task completion delay and economic expenditure for all the UEs. This objective is achieved through various strategies, including pre-installing new service instances into UAVs, removing idle service instances from UAVs, task offloading decision making, access control, selecting appropriate service instances for each offloaded service request, and resource allocation optimization. Considering the complexity of the problem and the dynamics of the system, we reformulate the problem as a Markov decision process (MDP) and present a Multi-Agent Deep Deterministic Policy Gradient (MADDPG)-based algorithm to enable low-complexity and real-time adaptive decision-making. Since our problem contains integer, binary and continuous variables, it is not straightforward to apply the MADDPG algorithm. Specifically, we first normalize the continuous variables, and then convert the continuous output generated by MADDPG into discrete variables, while ensuring the coupling constraints between different variables are preserved. The simulation results demonstrate the fast convergence of our proposed algorithm and its superior performance in minimizing costs compared with the baseline algorithms.**

***Index Terms*—Air-Ground Integrated Networks, computation offloading, service deployment, resource allocation, deep reinforcement learning.**

I. Introduction

With the rapid advancement of wireless communications, terrestrial wireless networks are experiencing an unprecedented surge in traffic, resulting in overwhelming congestion and an increased risk of network collapse [1]. In recent years, the frequency of natural disasters, such as earthquakes and tsunamis, has significantly increased due to the degradation of the natural environment. In such critical situations, the swift establishment of a temporary communication network is of utmost importance [2] [3]. Furthermore, in mission-critical scenarios like forest fire monitoring, in remote areas, or ensuring on-site communication during major holiday gatherings, it's not practical to deploy ground communication infrastructures [4]. On the other hand, smart user equipment (UE) has become an indispensable tool in our daily life, and the Internet of things (IoT) devices have also become ubiquitous. Many applications running on smart UEs and IoT devices (which are collectively referred to UEs) today are computation-intensive and resource-demanding [5], bringing great challenges to the processing capability and battery life of UEs [6] [7]. Multi-access edge computing (MEC) [8] and Air-Ground Integrated Networks (AGINs), have been considered for addressing those challenges. Leveraging MEC and Unmanned Aerial Vehicles (UAVs) for various IoT networks and wireless communication networks has emerged as a thriving research area in recent years.

MEC has been proposed as an efficient supplement to cloud computing [9]. By deploying computation and caching resources at edge nodes, MEC can empower UEs to run computation-intensive and resource-hungry applications, thereby reducing the task processing delay and the energy consumption [10]. In recent times, UAVs have emerged as valuable and practical tools for swiftly and cost-effectively deploying communication infrastructures. By equipping UAVs with MEC servers, UEs can benefit from reliable communication and efficient task processing services in various scenarios. Consequently, the research on MEC-empowered AGINs has become a prominent and dynamic area of study in recent years.

However, when combining MEC with UAVs, certain challenges need to be addressed. i) While many studies primarily address task offloading process-related issues, it is crucial to note that the deployment of task applications on UAV edge servers should be given precedence. Only after deploying the task applications on UAV edge servers can the tasks be successfully offloaded to them. ii) Given limited caching

*This work was supported in part by the Natural Science Foundation of China under Grant 62271391, in part by the Serving Local Special Scientific Research Project of Education Department of Shaanxi Province under Grant 21JC032, in part by the Horizon Europe Research and Innovation Program under grant 101086219, EPSRC grant EP/X038971/1, and Guangdong Province Science and Technology Project 2023A0505050127.

Jianbo Du, Ziwen Kong, and Aijing Sun are with School of Communications and Information Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, China. (Email: dujianboo@163.com; kzw15877611325@163.com; sunaijing@xupt.edu.cn)

Jiawen Kang is with the School of Automation, Guangdong University of Technology, China. (E-mail: kavinkang@gdut.edu.cn)

Dusit Niyato is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. (E-mail: dniyato@ntu.edu.sg)

X. Chu is with Department of Electronic and Electrical Engineering, The University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK. (Email: x.chu@sheffield.ac.uk).

F. Richard Yu is with Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada. (E-mail: Richard.Yu@carleton.ca).

capabilities of UAVs [11], deploying all task applications on each individual UAV is not feasible. Therefore, it is imperative to design optimized service placement strategies [12] thus to minimize the service distance between UEs and their serving UAVs. iii) Each UE has an option to either address its task locally by itself, or offload the task to a connected UAV. However, given that a UE might be covered by multiple UAVs, the question arises: should the task be offloaded or not? If the decision is made to offload the task, determining the optimal UAV for the UE to access becomes crucial. iv) When a UE is admitted and served by a UAV, the UAV has two options: it can either serve the UE with an available idle service instance that has already been deployed on the UAV, or it can initialize a new service instance specifically to process the UE's offloaded task. In this scenario, determining the best choice for the UAV is important. v) When multiple UEs offload tasks to the same UAV, then the UAV needs to allocate its available computation resources among the UEs in a thoughtful manner. Additionally, each UE should carefully design its transmit power given specific objectives, such as delay energy consumption and task processing cost. vi) In UAV-based systems, the environment is typically highly dynamic, and the objectives are often defined with long-term considerations. Given these factors, it becomes essential to meticulously develop algorithms that optimize the objectives while maintaining low complexity and delivering high-performance outcomes. Driven by the aforementioned rationales, our main contributions are given as follows.

- We present a novel MEC-based AGIN that facilitates the provision of Quality-of-Service (QoS)-guaranteed task processing services. Specifically, we investigate and optimize the whole process for task offloading. To achieve this objective, we first consider the service deployment process, including service instance initializing and removing, and then we conduct corresponding optimization for task offloading.
- Our objective is to minimize the long-term averaged weighted sum of task completion delay and task processing economic expenditure of the system. For this purpose, we jointly optimize task placement and replacement strategies during the service deployment phase, as well as decision making, access control, service instance selection, and resource allocation during the task offloading phase. Additionally, we ensure that the optimization process meets task processing QoS requirements, resource constraints, service instance placement and replacement constraints, load balancing, and other relevant factors.
- The complexity of the formulated problem is amplified by various factors, such as the mixed integer and nonlinear properties, real-time dynamics of wireless channel gains, time-varying locations of UAVs and UEs, fluctuating communication and computation resources, and random user requests. Traditional convex optimization and dynamic programming approaches struggle to solve the problem effectively. To address this, we reframe the problem as a Markov Decision Process (MDP) and propose a Deep Reinforcement Learning (DRL) based algorithm utilizing the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) approach. However, since MADDPG is designed for continuous problems [13], we reformulate the output of MADDPG to handle integer and binary variables, enabling us to obtain solutions for these types of variables as well.

The subsequent sections are structured as follows. In section II, we provide an overview of the related works. Section III introduces the system model, while Section IV delves into our problem formulation. In Section V, we reformulate the problem as a MDP and propose a joint optimization algorithm based on DRL. In Section VI, our simulation results are presented and discussed. Finally, our paper is concluded in Section VII, summarizing the key findings and contributions.

II. Related Works

Recent years, there has been a remarkable amount in research on MEC empowered AGINs, which has emerged as a prominent and highly active research area. In most existing studies, MEC-enabled UAVs usually act as the edge computing node, and will provide UEs with computation offloading (or task offloading) services.

Reference [14] addressed the computation offloading challenges in UAV and MEC-assisted IoT systems. Their approach intended to optimize user access control, UAV positioning, and resource allocation of the UAV edge server, etc., to minimize the maximum task processing latency among all end devices. In [15], a load-balancing based dynamic entry and exit mechanism was introduced within a multi-UAV edge computing framework, and thus to ensure the seamless delivery of uninterrupted, high-quality task offloading services to users situated within the designated area. To address the objective of minimizing the maximum latency in task processing across all UEs, reference [16] proposed to optimize service and UAV placement, and task offloading decision-making. In [17], the authors explored the utilization of a UAV-assisted MEC system to optimize platoon moving vehicles. Their objective was to maximize the energy efficiency of MEC services through radio and computation resource scheduling optimization. To cater to the service requirements of remote edge users, the authors in [18] introduced a multi-hop task offloading framework where UAVs work collectively to enable multi-hop task offloading through joint resource allocation and deployment optimization. The aforementioned studies employed traditional optimization algorithms to address their problems. However, traditional methods often exhibit high complexity, which limit their efficiency in finding optimal solutions. Additionally, when considering the real-time dynamics of the network, traditional methods may struggle to provide solutions that can adapt to changing conditions effectively.

Several studies have incorporated intelligent algorithms [19] AGIN networks. In the paper by [20], the authors integrated UAVs, edge computing, etc., into IoT networks to facilitate data transmission in impaired machine-to-machine communication networks. The authors intended to maximize the data computation capacity, and presented an adaptive algorithm based on dueling Deep Q-network (DQN) to obtain optimal strategies. In [21], the authors focused on task

scheduling issues within space-air-ground integrated networks enabled IoT systems, and presented an Actor-Critic (AC) based optimization algorithm to effectively allocate computing resources among different virtual machines deployed at UAV edge servers and to devise task scheduling strategies. In [22], the authors tackled the task offloading challenge within a blockchain-enabled space-air-ground integrated IoT system. They primarily concerned optimizing UAV selection for access, offloading decision making between the UAVs and the satellite, and allocating resources between blockchain tasks and general computing tasks, in order to minimize the queuing delay in the system. In [23], the authors delved into the task offloading related challenges within a three-layer space-air-ground heterogeneous network to maximize the total transmit rate of all ground UEs. To achieve this, they employed a combination of multi-agent proximal policy optimization algorithm, Lyapunov optimization, and convex optimization techniques, thus to optimize task scheduling, high-altitude platform selection, etc. The authors in [24] studied the user association and computation resource allocation problem in an UAV assisted vehicular networks, where the macro eNodeB and UAV act as access points and edge nodes. Based on MADDPG, the number of offloaded tasks is maximized, and the heterogeneous QoS is satisfied. The aforementioned studies primarily focused on the task offloading process, assuming that the services have already been deployed on UAVs. However, they did not consider scenarios where the required application is not available on the edge servers, that require task should first be placed and then can task offloading be conducted.

In [25], the authors investigated a MEC-empowered IoT system and specifically focused on the service caching and placement issues for IoT tasks driven by sensed data, and they proposed a DRL based algorithm to get the optimal solution. The research presented in reference [26] focuses on the Virtual Network Function (VNF) deployment problem within a satellite-facilitated MEC system. This problem is formulated as a potential game, and the authors proposed an iterative algorithm to obtain a suboptimal VNF deployment strategy to adopt to user requests. While the works mentioned in [25] and [26] may provide valuable insights into task placement strategies and resource allocation, they do not consider the dynamic decision-making process of task offloading.

Few studies have comprehensively examined the complete task offloading process in AGIN, encompassing both task placement and offloading procedures, so as for joint optimization. Reference [27] proposed an MEC enabled space-air-ground integrated framework, where a low earth orbit satellite serves as the edge server, and they investigated joint caching placement and offloading decision for the vehicles in remote areas, while other related issues mentioned in offloading, such as resource allocation, power control, etc., are not considered. In [28], the authors conducted a comprehensive investigation into computation offloading issues in an edge computing enabled ultra-dense network. Their study focused on addressing the joint challenges of service placement, task partitioning strategy, etc., to minimize task processing delays and enhance network resource utilization. Then the authors proposed a two-layer DQN based algorithm. The upper layer determined delay-insensitive cache placement decisions in a slower timescale, and the lower layer operated in a faster timescale and generated delay-sensitive computation offloading and resource allocation decisions. Reference [28] mainly considers task offloading issues in ground scenarios. In air-ground scenarios, there are several other crucial factors that need to be considered, such as access control, etc. Furthermore, it is important to address the challenge of task replacement when the storage space of the UAV becomes full because of the limited storage volume of UAVs.

Inspired by the aforementioned considerations, we present a novel approach by jointly examining the entire process of task offloading in an AGIN that encompasses the optimization of both the task deployment process and the task offloading process. In task deployment process, we consider task placement for newly arrived task requests, and the task replacement for the placed task when the storage space is full. We consider that each service instance can only serve one UE, and we should decide whether the task should be served by a new or an existing service instance. In the task offloading process, additional considerations need to be taken into account. Firstly, we need to determine each UE should select local processing or task offloading modes, and which specific UAV that the UE should access in task offloading mode. Furthermore, it is essential to consider the data transmit power of each UE in computation offloading, and the resource allocation of each UAV when it serves multiple UEs. Given the complexity of the problem and the highly dynamics of the environment, tackling the challenges becomes inherently difficult. In the subsequent sections, we offer an MADDPG based algorithm to tackle the problem for adaptive real-time solutions, and intend for long-term system performance optimization. Since our problem contains binary, integer and continuous variables, while MADDPG can only solve problems with continuous variables, we will improve MADDPG to adapt to our problem.

III. System Model

In Fig. 1, we present an overview of the considered AGIN, which comprise a collection of UAVs equipped with MEC capabilities, as well as a set of UEs. The set of UAVs and UEs are represented as $\mathcal{J} = \{1, 2, ..., j, ..., J\}$ and $\mathcal{I} = \{1, 2, ..., i, ..., I\}$, respectively, and the symbols J and I stand for the total numbers of UAVs and UEs. Each UAV is assigned an j within the set $\mathcal{J}$, and each UE is assigned an index i within the set $\mathcal{I}$. In our system, each UAV and the edge server deployed on it are treated as the same entity and can be referred to as either UAV j or edge server j, and UAVs have the ability to establish direct communication links with each other via wireless connections. The system operates in a time-slotted manner, where each time slot (or time step) represents a discrete unit of time for task placement, offloading and processing. We use $\mathcal{T} = \{1, 2, ..., t, ..., T\}$ to represent the set of time slots, where T denotes the number of time slots in the system. The duration of each time slot is τ. At the beginning of each time slot, each UE has a compute-intensive task to be processed, which can either be tackled by the UE itself, or be offloaded to a UAV for stronger task executing capability at

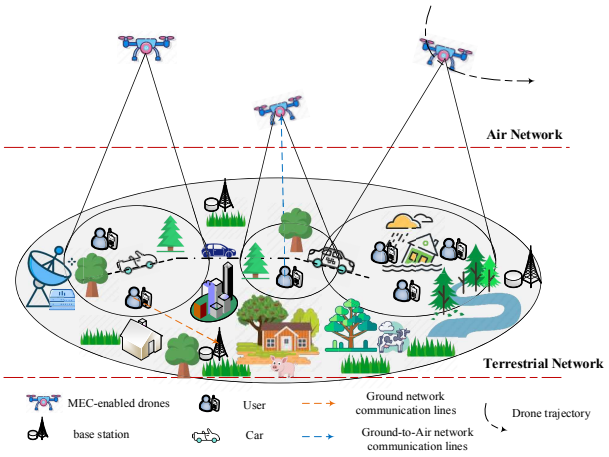


Fig. 1: The concerned scenario.

a certain cost. To minimize the delay in task processing, the effective approach is to pre-install service instances on UAVs before the arrival of user requests. Idle existing instances can also be removed to make room for deploying other instances for frequently requested services. Especially, we call UAV 1 as the primary UAV, which has the most powerful processing capabilities and plenty energy. Other UAVs are referred to as general UAVs, whose computing power and energy supply are relatively limited.

A. Task Model

In the given scenario, there exist H distinct types of services, where the set of tasks is represented by $\mathcal{H} = \{1, 2, ..., h, ..., H\}$ [29]. Each task h is described by the parameters $\Lambda_h = \{D_h, \Xi_h, O_h, w_h, T_h^{max}\}$. Here, D_h (measured in bits) denotes the input data size associated with the task; Ξ_h (measured in CPU cycles/s) represents the task processing density, i.e., the number of CPU cycles necessary to address one bit of the task; based on D_h and Ξ_h, the computation amount C_h can be given by $C_h = D_h \Xi_h$; O_h (measured in bits) specifies the size of the result obtained from processing the task, which is considerably smaller than the input data size D_h [9]; w_h (measured in bits) refers to the size of a service instance for the specific service h; T_h^{max} (measured in seconds) indicates the maximum acceptable delay or latency for completing the task, and it defines the time constraint within which the task should be finished.

B. Service placement stage

In this section, we investigate the real-time service placement (or service deployment, task placement, task deployment) strategies in the system. Service (or task) is an abstraction of applications that can be deployed at UEs and UAVs, and requested by UEs. To run a particular service at a device (i.e., UAVs and UEs in this paper), the device has to deploy the service, i.e., to store the data associated with the services, such as libraries and databases. As such, we consider that each UE deploys its required service and can process its required task itself, so we do not consider the task placement issues at the UE end. The UAVs also have certain storage space for service deployment. However, the caching capacity of the UAVs are limited, and each UAV could not deploy all services. In this paper, we adopt the assumption that each instance of a task deployed at a UAV is capable of serving only one task processing request from only one UE [30].

During the service placement stage, we have the flexibility to deploy or remove service instances from UAVs. We denote the variable $\delta_{h,j}(t)$ as the number of instances of task h that will be installed or removed from UAV j at time slot t. If $\delta_{h,j}(t) > 0$, it indicates that $\delta_{h,j}(t)$ instances of service h will be deployed at UAV j; If $\delta_{h,j}(t) < 0$, it means that $\delta_{h,j}(t)$ instances of service h will be released from UAV j; and if $\delta_{h,j}(t) = 0$, it implies that no instances of service h will be placed or deleted from UAV j at that particular time step. Finally, we adapt variable $n_{h,j}(t)$ to represent the number of instances of service h at UAV j after the service placement stage at time slot t. Note that the instances of a same task h are indistinguishable, once the number $\delta_{h,j}(t)$ is determined, $\delta_{h,j}(t)$ random instances will be removed from the UAV j.

C. Task Offloading Stage

At the start of each time slot t, every UE i creates a task from the H tasks. The task request generated by UE i is represented by the binary variable $r_{i,h}(t) \in \{0, 1\}$. Specifically, if $r_{i,h}(t) = 1$, it indicates that UE i generates task h, otherwise $r_{i,h}(t) = 0$. For each task, there are two possible processing options: local processing or offloading to a UAV. The decision of whether to offload the task or process it locally is denoted by the binary variable $\varrho_i(t) \in \{0, 1\}$ for UE i. If $\varrho_i(t) = 1$, it indicates the task of UE i will be offloaded to a UAV for processing. Conversely, if $\varrho_i(t) = 0$, it indicates that UE i will process the task locally by itself.

1) Local Processing: Define f_i^l (measured in CPU cycles/s) as the local CPU frequency of UE i, and the delay of processing task h locally at UE i can be given by

$$T_i^l(t) = \frac{\sum_{h \in \mathcal{H}} r_{i,h}(t) C_h(t)}{f_i^l}. \tag{1}$$

The expense of local processing comes from the energy consumption, which can be given by

$$Exp_i^l(t) = \eta^l f_i^l, \tag{2}$$

where η_i^l (measured in \$/(CPU cycles/s)) denotes the price in using UE i's local CPU for task processing.

2) UAV Edge Processing: If UE i intends to offload its task to a UAV j, then the overall incurred delay consists of three components: data delivery delay, which represents the time that it takes to transmit the input data from UE i to UAV j; task processing delay at the UAV, which accounts for the time taken by the UAV to perform the task and generate the task processing result; result transmission delay, which means the time taken to transfer the processing result from UAV j to UE i. Considering that the size of task processing results is typically very tiny, we can omit the result delivery latency in UAV edge processing mode. In addition to delay considerations, it is important to account for the economic expenditure of UEs in task offloading decision making, such

as the monetary price of using UAV resources or other relevant factors.

In order to obtain the delay and economic expenditure of edge processing mode, we first need to consider the data rate of ground-UAV wireless links. The path loss between UE i and its associated UAV j could be expressed as [23]

$$PL_{i,j}(t) = 20\log\left(\frac{4\pi f_c\sqrt{x_{i,j}^2(t)+y_j^2(t)}}{c}\right) + pro_{i,j}^{LoS}(t)\epsilon_{i,j}^{LoS}(t) + (1-pro_{i,j}^{LoS}(t))\epsilon_{i,j}^{NLoS}(t), \quad (3)$$

where $x_{i,j}(t)$ (measured in meters) expresses the horizontal distance between UE i and UAV j, $y_j(t)$ (in meters) represents the vertical altitude of UAV j from the ground, f_c (measured in Hertz) corresponds to the carrier frequency, and c (measured in m/s) stands for the speed of light. In addition, $\epsilon_{i,j}^{LoS}(t)$ and $\epsilon_{i,j}^{NLoS}(t)$ denotes the additional path loss experienced beyond the free space path loss due to Line-of-Sight (LoS) and Non-Line-of-Sight (NLoS) transmission, respectively. Moreover, the probability of whether the LoS link exists between ground UE i and UAV j can be expressed as [23]

$$pro_{i,j}^{LoS}(t) = \frac{1}{1+\alpha_1\exp\{-\alpha_2[\arctan(\frac{y_j(t)}{x_{i,j}(t)})-\alpha_1]\}}, \quad (4)$$

where the symbols α_1 and α_2 are constant parameters.

Denote the bandwidth of each UAV j as $B_j,\ j\in\mathcal{J}$, which will be shared among the UEs that are served by UAV j in Non-Orthogonal Multiple Access (NOMA), thus to achieve better channel utilization, thereby obtaining higher channel transmission rates and larger system capacity. Define $a_{i,j}(t),\ i\in\mathcal{I},\ j\in\mathcal{J}$ as the access control scheme of UE i. In this context, $a_{i,j}(t)=1$ indicates that UE i selects UAV j for task offloading in time step t, and $a_{i,j}(t)=0$ otherwise. Then, the wireless data delivery rate between the ground UE and its attached UAV could be expressed as

$$R_{i,j}(t) = B_j\log\left(1+\frac{a_{i,j}(t)p_{i,j}(t)h_{i,j}(t)}{\sum\limits_{l\in\mathcal{I},h_{l,j}(t)<h_{i,j}(t)} a_{l,j}(t)p_{l,j}(t)h_{l,j}(t)+\sigma^2}\right), \quad (5)$$

where $h_{i,j}(t) = 10^{\frac{-PL_{i,j}(t)}{10}}$, $p_{i,j}(t)$ represents the transmit power when UE i offloads data to UAV j, and σ^2 denotes the noise power.

Next, we discuss the delay and economic expenditure in UAV edge processing mode. For a UAV j, in order to accommodate the task processing request $r_{i,h}(t)$ from UE i, two conditions must be met: either there should be an available idle service instance at UAV j, or there should be sufficient space to initialize a new instance for task h. We introduce a binary variable $a_{i,j,h}^{idle}(t)$ to represent the scenario where the task processing request $r_{i,h}(t)$ is offloaded to UAV j and served by an idle instance of task h, and use $a_{i,j,h}^{new}(t)$ to indicate the case when the request $r_{i,h}(t)$ is served by a newly deployed instance of task h at UAV j. Consequently, we have

$$a_{i,j}(t)r_{i,h}(t) = a_{i,j,h}^{idle}(t) + a_{i,j,h}^{new}(t). \quad (6)$$

For each UE i, the delay to complete the task execution in UAV edge processing mode consists of the service deployment delay (if its request is served by a newly initialized service instance), the data transmission delay, and the task processing delay, which can be expressed as

$$T_i^u(t) = \sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{H}}\left[a_{i,j,h}^{idle}(t)\left(\frac{r_{i,h}(t)D_h(t)}{R_{i,j}(t)}+\frac{r_{i,h}(t)C_h(t)}{f_{i,j}(t)}\right) + a_{i,j,h}^{new}(t)\left(D_{h,j}^{place}+\frac{r_{i,h}(t)D_h(t)}{R_{i,j}(t)}+\frac{r_{i,h}(t)C_h(t)}{f_{i,j}(t)}\right)\right], \quad (7)$$

where $f_{i,j}(t)$ (measured in CPU cycles/s) represents the computation resource assigned to UE i by UAV j, and $D_{h,j}^{place}$ (measured in second) is the delay of initializing a service instance for task h on UAV j.

The economic expenditure of UAV edge processing mode arises from two components, i.e., the utilization of the wireless channel for data transmission in task offloading, and the consumption of computation resources for task processing, which can be given by

$$Exp_i^u(t) = \sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{H}}\left[a_{i,j,h}^{idle}(t)(\eta^{j,1}R_{i,j}(t)+\eta^{j,2}f_{i,j}(t)) + a_{i,j,h}^{new}(t)(\eta^{j,1}R_{i,j}(t)+\eta^{j,2}f_{i,j}(t)+\lambda_{h,j})\right], \quad (8)$$

where $\eta^{j,1}$ (measured in \$/bps) represents the price for utilizing the wireless channel resource of UAV j, $\eta^{j,2}$ (measured in \$/(CPU cycles/s)) denotes the price for utilizing the computation resources of UAV j, and $\lambda_{h,j}$ (in \$) is the additional economic cost for using a newly deployed instance of service h on UAV j, respectively.

Combining local and edge processing together, the total delay for completing the task for UE i is

$$T_i(t) = (1-\varrho_i(t))T_i^l(t) + \varrho_i(t)T_i^u(t), \quad (9)$$

and the economic expenditure of UE i is expressed as

$$Exp_i(t) = (1-\varrho_i(t))Exp_i^l(t) + \varrho_i(t)Exp_i^u(t). \quad (10)$$

Considering both the delay and economic expenditure factors, the cost of each UE i is defined as

$$Cost_i(t) = \gamma^1\vartheta^t T_i(t) + \gamma^2\vartheta^c Exp_i(t), \quad (11)$$

where $\gamma^1\in[0,1]$ and $\gamma^2\in[0,1]$ are weight coefficients which determine the importance of reducing delay and reducing economic expenditure, respectively, and we have $\gamma^1+\gamma^2=1$. The two parameters ϑ^t (in Cost/s) and ϑ^c (in Cost/\$) are coefficients for balancing the range of delay and economic expenditure to the same magnitude, and also for unifying the unit of the two quantities. The important notations are summarized in Table I.

IV. Problem Formulation

A. Problem Formulation

In this paper, our objective is to minimize the long-term average system cost by jointly optimizing the task placement $\mathbf{\Delta}(t)=\{\delta_{h,j}(t)\},\ h\in\mathcal{H},\ j\in\mathcal{J}$, the offloading decision $\boldsymbol{\varrho}(t)=\{\varrho_i(t)\},\ i\in\mathcal{I}$, the access control and instance selection $\mathbf{A}(t)=\{a_{i,j}(t),a_{i,j,h}^{idle}(t),a_{i,j,h}^{new}(t)\},\ i\in\mathcal{I},\ j\in\mathcal{J},\ h\in\mathcal{H}$, the transmit power control $\mathbf{P}(t)=\{p_{i,j}(t)\},\ i\in\mathcal{I},\ j\in\mathcal{J}$, and the computation resource assignment $\mathbf{F}(t)=$

$\{f_{i,j}(t)\},\ i \in \mathcal{I},\ j \in \mathcal{J}$. The joint optimization problem is formulated as

$$(\mathcal{P}_1): \min_{\boldsymbol{\Delta}(t),\boldsymbol{\varrho}(t),\mathbf{A}(t),\mathbf{P}(t),\mathbf{F}(t)} \left[\frac{1}{T^{max}} \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{I}} Cost_i(t)\right] \quad (12)$$

$$\text{s.t. } (\text{C1}): \ \delta_{h,j}(t) \in \mathbb{N},\ h \in \mathcal{H},\ j \in \mathcal{J},$$

$$(\text{C2}): \ n_{h,j}(t-1) \geq |\delta_{h,j}(t)| \cdot \mathbb{I}\{\delta_{h,j}(t) < 0\}, \quad h \in \mathcal{H},\ j \in \mathcal{J},$$

$$(\text{C3}): \ n_{h,j}(t-1) + \delta_{h,j}(t) \cdot \mathbb{I}\{\delta_{h,j}(t) > 0\} \leq thres, \quad h \in \mathcal{H},\ j \in \mathcal{J},$$

$$(\text{C4}): \ \varrho_i(t) \in \{0,1\},\ i \in \mathcal{I},$$

$$(\text{C5}): \ a_{i,j}(t) \in \{0,1\},\ i \in \mathcal{I},\ j \in \mathcal{J},$$

$$(\text{C6}): \ a^{idle}_{i,j,h}(t), a^{new}_{i,j,h}(t) \in \{0,1\}, \quad i \in \mathcal{I},\ j \in \mathcal{J},\ h \in \mathcal{H},$$

$$(\text{C7}): \ \sum_{j\in\mathcal{J}} a_{i,j}(t) \cdot \mathbb{I}\{\varrho_i(t) = 1\} = 1,\ i \in \mathcal{I},$$

$$(\text{C8}): \ a^{idle}_{i,j,h}(t) + a^{new}_{i,j,h}(t) = a_{i,j}(t) r_{i,h}(t), \quad i \in \mathcal{I},\ j \in \mathcal{J},\ h \in \mathcal{H},$$

$$(\text{C9}): \ \sum_{h\in\mathcal{H}} n_{h,j}(t) w_h \leq \Pi_j,\ j \in \mathcal{J},$$

$$(\text{C10}): \ \sum_{i\in\mathcal{I}} \varrho_i(t) a_{i,j}(t) r_{i,h}(t) \leq n_{h,j}(t),\ j \in \mathcal{J},$$

$$(\text{C11}): \ \sum_{j\in\mathcal{J}} \varrho_i(t) a_{i,j}(t) = 1,\ i \in \mathcal{I},$$

$$(\text{C12}): \ 0 < p_{i,j}(t) < p^{max},\ i \in \mathcal{I},\ j \in \mathcal{J},$$

$$(\text{C13}): \ 0 < f_{i,j}(t) < f^{max}_j,\ i \in \mathcal{I},\ j \in \mathcal{J},$$

$$(\text{C14}): \ \sum_{i\in\mathcal{I}} f_{i,j}(t) \leq f^{max}_j,\ j \in \mathcal{J},$$

$$(\text{C15}): \ T_i(t) \cdot \mathbb{I}\{r_{i,h}(t) = 1\} \leq T^{max}_h,\ i \in \mathcal{I},\ j \in \mathcal{J},$$

where $thres$ is a threshold for the number of deployed service instances in task placement; $\Pi_j,\ j \in \mathcal{J}$ represents the storage capacity of the UAV j for application data caching in task placement; p^{max} denotes the maximum transmit power of UEs; f^{max}_j represents the processing capability of MEC server j; T^{max}_h refers to the maximum tolerable task processing latency of task h; and additionally, $\mathbb{I}\{\cdot\}$ is the symbolic function, where $\cdot$ holds when $\mathbb{I}\{\cdot\} = 1$, and otherwise, $\mathbb{I}\{\cdot\} = 0$.

In $(\mathcal{P}_1)$[1], (C1) is the integer requirement on task instance placement; (C2) is used to assure that the removed number of instances of each service h should be less than the number of instances deployed on UAV j; (C3) is used to prevent too many instances are installed at a certain UAV, and thus to keep load balancing among UAVs; (C4), (C5), and (C6) represent the binary constraints related to task offloading decision, access control, and service instance selection; (C7) assures that one UE can only access one UAV; (C8) assures that there must be a deployed instance in UAV j to serve its admitted UE i; (C9) ensures that the volume of the deployed tasks should be less than the storage volume of each UAV; (C10) constrains that the number of UEs that admitted in UAV j for processing task h must not exceed the number of instances of task h in UAV j; (C11) constrains that each UE should offload its task to only one UAV; (C12) represents the constraint on transmit power control; (C13) requires that the computation resource allocation must be non-negative; (C14) imposes constraints that the assigned computation resource must be less than the processing capability of each UAV; finally, (C15) is the QoS requirement, which ensures that the task should be completed within the maximum tolerable delay.

[1]Please note that we omit the $t \in \mathcal{T}$ in each constraint for concise.

Remark 1: *We consider that the duration τ of each time slot should be greater than or equal to the maximum tolerable latency of all tasks, as specified in equation (C14). This assumption helps avoiding the challenges of dealing with issues of queues [31]. The exploration of cross-slot optimization will be considered as a part of our future work.*

Remark 2: *We assume all UAVs and UEs work in an ideal synchronization state.*

The formulated problem $(\mathcal{P}_1)$ is a mixed-integer and non-linear programming (MINLP) problem, and it is known to be generally NP-hard and therefore challenging to solve optimally. Moreover, the decision-making process in this problem occurs in a highly dynamic environment for long-term optimization, which makes it challenging for traditional convex optimization algorithms to adapt to the unknown environment and perform adaptive optimization. DRL algorithms are designed to optimize a long-term objective [32], and we attempt to exploit DRL to solve our problem. Since our problem contains integer, binary, and continuous variables, discrete control oriented DRL algorithms, such as DQN, AC, and advantage actor-critic (A2C), could not work well [33] [34]. Although we can resort to discretization, we have to face the curse of dimension issues and dramatically deteriorating performance. Deep deterministic policy gradient (DDPG) is specifically designed to address continuous control problems, so it is still not appropriate for solving our formulated mixed integer and continuous problem. However, we can reshape DDPG for our problem, and propose a more suitable joint optimization algorithm based on MADDPG for MINLP, and thus for more reliable, real-time, and high efficient system control and decision making optimization.

V. Joint optimization Algorithm Based on MADDPG for our MINLP Problem

In this section, we demonstrate the utilization and improvement of MADDPG to learn real-time control strategies of our problem. We begin by introducing the preliminary concepts of MADDPG. Subsequently, we reformulate our problem $(\mathcal{P}_1)$ as a MDP, by constructing its state space, action space, immediate reward, and state transformation for each agent. Next, we present the framework of our MADDPG based optimization algorithm for MINLP problem.

A. Preliminaries of MADDPG

In single-agent DRL frameworks [35], the agent strengthens its decision-making abilities by continuously and actively interacting with the environment [36]. Specifically, DDPG algorithm [34] belongs to the DRL algorithm family, that is well-suited for continuous control tasks. MADDPG is introduced as

an extension of DDPG, which aims at addressing the common challenge in complex systems where individual agents are often faced with limited observations, which significantly hampers their learning process. MADDPG operates under the principle of centralized training and decentralized executing. In this framework, every agent is associated with an actor and a critic, referred to as $\mu_j(o_j(t);\theta_j)$ and $q_j(\mathbf{s}(t),\mathbf{a}(t);w_j)$, respectively. Each actor network $\mu_j(o_j(t);\theta_j)$ works in a decentralized and deterministic manner, i.e., for a given input $o_j(t)$, it will output a deterministic action $\mathbf{a}_j(t) = \mu_j(o_j(t);\theta_j)$. For each critic network $q_j(\mathbf{s}(t),\mathbf{a}(t);w_j)$, its input includes the global state $\mathbf{s}(t) = \{o_1(t),...,o_J(t)\}$ and action $\mathbf{a}(t) = \{\mathbf{a}_1(t),...,\mathbf{a}_J(t)\}$, which comprises the observations and actions of all agents at time t. The output of the critic network is a real number that serves as an estimation of the performance associated with taking action $\mathbf{a}(t)$ under the given state $\mathbf{s}(t)$. The actor network of the jth agent $\mu_j(o_j(t);\theta_j)$ is used to control its corresponding agent j, while the jth critic network $q_j(\mathbf{s}(t),\mathbf{a}(t);w_j)$ is used for evaluating action $\mathbf{a}(t)$, and the score its gives can be used to guide the jth actor network to train its parameters, and thus to make better actions. In order to cut off "bootstrapping" to avoid error propagation, thereby alleviating the overestimation issues of MADDPG, target networks are employed for calculating time difference (TD) targets. For each actor network, there's a corresponding target actor network, which is denoted as $\mu_j(o_j(t);\theta_j^-)$; also, each critic network corresponds to a target critic network $q_j(\mathbf{s}(t),\mathbf{a}(t);w_j^-)$. Each target network in MADDPG shares the same structure as its corresponding primary actor or primary critic, but with different parameters. The parameters of the jth primary actor and primary critic are represented by θ_j and w_j, and correspondingly, the parameters of the jth target actor and target critic are represented by θ_j^- and w_j^-, respectively.

We first discuss the training process of actor networks. In the training process, actor networks are trained using deterministic policy gradient, and critic networks are trained employing TD algorithm. As an off-policy DRL algorithm, MADDPG leverages experience replay to mitigate temporal correlations among training samples, enabling the reuse of past experiences for network training. This mechanism not only helps in stabilizing the training, but also promotes more efficient exploration and thus improves the overall performance of the algorithm. Experiences are represented as tuples capturing state, action, reward, and next state information, with the form $<\mathbf{s}(t),\mathbf{a}(t),\mathbf{r}(t),\mathbf{s}(t+1)>$, $t \in \mathcal{T}$. MADDPG utilizes a centralized replay buffer, which is located in the central controller and serves as a repository for all the experiences collected during the training process. In a general experience, $\mathbf{s}(t)$ and $\mathbf{a}(t)$ are the global state and the actions taken by all agents at time step t as aforementioned, respectively. Similarly, $\mathbf{r}(t) = \{r_1(t),...,r_J(t)\}$ signifies the rewards received by each agent at time step t, and $\mathbf{s}(t+1) = \{o_1(t+1),...,o_J(t+1)\}$ is the new global state observed by all agents in time step $t+1$, respectively.

The purpose of training the jth actor network $\mu_j(o_j(t);\theta_j)$ is to optimize its parameter θ_j, thus to improve the score given by the jth critic network. The objective function of the jth actor network in MADDPG can be defined as

$$\begin{aligned}&\mathfrak{J}_j(\theta_1,...,\theta_J) \\ &= \mathbb{E}_S\Big[q\Big(S,\big[\mu_1(O_1(t);\theta_1),...,\mu_J(O_J(t);\theta_J)\big];w_j\Big)\Big],\end{aligned} \tag{13}$$

where the expectation is about state $S = [O_1,...,O_J]$. The gradient of the objective function with respect to the parameters of the jth actor can be computed by

$$\begin{aligned}&\nabla_{\theta_j}\mathfrak{J}_j(\theta_1,...,\theta_J) \\ &= \mathbb{E}_S\Big[\nabla_{\theta_j}q\Big(S,\big[\mu_1(O_1(t);\theta_1),...,\mu_J(O_J(t);\theta_J)\big];w_j\Big)\Big].\end{aligned} \tag{14}$$

Next, we adopt the Monte Carlo method to appropriate the expectation in the above equation. We sample a batch of M experience from the experience pool, where a certain experience m can be given by $<\mathbf{s}^m(t),\mathbf{a}^m(t),\mathbf{r}^m(t),\mathbf{s}^m(t+1)>$. For each state $\mathbf{s}^m(t) = [o_1^m(t),...,o_J^m(t)]$, we can obtain J actions by the J actor networks, which can be given by

$$\begin{aligned}\hat{\mathbf{a}}_1^m(t) &= \mu_1(o_1^m(t);\theta_1), \\ &\vdots \\ \hat{\mathbf{a}}_J^m(t) &= \mu_J(o_J^m(t);\theta_J).\end{aligned} \tag{15}$$

The gradient of the objective function with respect to the parameters of the jth actor, i.e., θ_j, can be given by

$$\begin{aligned}\mathbf{g}_{\theta_j}^m &= \nabla_{\theta_j}q\Big(\mathbf{s}^m(t),\big[\mu_1(o_1^m(t);\theta_1),...,\mu_J(o_J^m(t);\theta_J)\big];w_j\Big) \\ &= \nabla_{\theta_j}q\Big(\mathbf{s}^m(t),\big[\hat{\mathbf{a}}_1^m(t),...,\hat{\mathbf{a}}_J^m(t)\big];w_j\Big) \\ &= \nabla_{\theta_j}\mu_j(o_j^m(t);\theta_j)\cdot\nabla_{\hat{\mathbf{a}}_j^m(t)}q\Big(\mathbf{s}^m(t),\big[\hat{\mathbf{a}}_1^m(t),...,\hat{\mathbf{a}}_J^m(t)\big];w_j\Big),\end{aligned} \tag{16}$$

and the average gradient is given by

$$\mathbf{g}_{\theta_j} = \frac{1}{M}\sum_{m=1}^{M}\mathbf{g}_{\theta_j}^m. \tag{17}$$

The parameter θ_j of actor j can be updated by gradient ascend as follows

$$\theta_j \leftarrow \theta_j + \beta\cdot\mathbf{g}_{\theta_j}. \tag{18}$$

Next, we discuss the training methods of critic networks. Each critic network $q_j(\mathbf{s}(t),\mathbf{a}(t);w_j)$ can be trained using TD algorithm, so as to fit the action value function $Q_j(\mathbf{s},\mathbf{a})$ as possible. Given an experience $<\mathbf{s}^m(t),\mathbf{a}^m(t),\mathbf{r}^m(t),\mathbf{s}^m(t+1)>$, and for the state $\mathbf{s}^m(t+1) = [o_1^m(t+1),...,o_J^m(t+1)]$, we can obtain J actions by the J target actor networks at time step $t+1$, which can be given by

$$\begin{aligned}\hat{\mathbf{a}}_1^{m-}(t+1) &= \mu_1(o_1^m(t+1);\theta_1^-), \\ &\vdots \\ \hat{\mathbf{a}}_J^{m-}(t+1) &= \mu_J(o_J^m(t+1);\theta_J^-),\end{aligned} \tag{19}$$

and we have

$$\hat{\mathbf{a}}^{m-}(t+1) = \Big[\hat{\mathbf{a}}_1^{m-}(t+1),...,\hat{\mathbf{a}}_J^{m-}(t+1)\Big]. \tag{20}$$

The loss function is

$$L(\theta_j) = \frac{1}{2M}\sum_{t=1}^{M}\Big(\hat{y}_j^m(t) - q(\mathbf{s}^m(t), \mathbf{a}^m(t); w_j)\Big)^2, \quad (21)$$

where $\hat{y}_j^m(t)$ is TD target, which is expressed as

$$\hat{y}_j^m(t) = r_j^m(t) + \gamma \cdot q\Big(\mathbf{s}^m(t+1), \hat{\mathbf{a}}^{m-}(t+1); w_j\Big). \quad (22)$$

Then, TD error can be given by

$$\delta_j^m(t) = q\Big(\mathbf{s}^m(t), \mathbf{a}^m(t); w_j\Big) - \hat{y}_j^m(t). \quad (23)$$

To update the parameters w_j of the jth critic network, gradient descent algorithm can be applied as follows

$$w_j \leftarrow w_j - \alpha \cdot \frac{1}{M}\sum_{t=1}^{M}\delta_j^m(t) \cdot \nabla_{w_j} q\Big(\mathbf{s}^m(t), \mathbf{a}^m(t); w_j\Big). \quad (24)$$

In the end, each agent j refreshes its target actor and critic networks as follows

$$\begin{aligned} \theta_j^- &\leftarrow \zeta \cdot \theta_j + (1-\zeta)\cdot\theta_j^-, \\ w_j^- &\leftarrow \zeta \cdot w_j + (1-\zeta)\cdot w_j^-, \end{aligned} \quad (25)$$

where $\zeta \in [0, 1]$.

B. Problem Reformulation

In the considered system, every UAV functions as a DRL agent to make independent and localized decisions based on its observations and learned policies. Besides, the primary UAV serves as the controller which possesses the global information of the whole system. At the start of each time slot, every UE initiates its task processing request and sends it to all UAVs it is associated with. Each UAV gathers the information, e.g., the task requests information, the dynamics of the network states, etc., from all its covered UEs, and then sends it to the centre controller at the primary UAV. After obtaining the global network state information, centralized training process will proceed at the centre controller. After training, each agent will perform distributed decision making based on its observation $o_j(t)$. To utilize DRL algorithms to address the formulated problem $(\mathcal{P}_\mathbf{1})$, it is necessary to convert it into the standard form of MDP. The key components of this transformation include defining the state space, action space, reward function, and state transition for each agent, which are given as follows.

1) State Space: At the start of each time step t, every agent $j,\ j \in \mathcal{J}$ gathers and obtains its own observation $o_j(t)$ from the environment. Please note that each UAV can only observe the dynamics of its covered UEs. Denote the set of UEs covered by UAV j as $\mathcal{I}_j$, where the distance between each UE i in $\mathcal{I}_j$ satisfies

$$\sqrt{x_{i,j}(t)^2 + y_j(t)^2} \le D_{th}, \quad (26)$$

where D_{th} is the distance threshold [3].

The observation of UAV j, i.e., $o_j(t)$, could be represented as

$$o_j(t) \triangleq \{\mathbf{r}_j(t),\ \mathbf{N}_j(t-1),\ \boldsymbol{\epsilon}_j(t),\ \mathbf{X}_j(t),\ \mathbf{Y}_j(t)\}, \quad (27)$$

where

- $\mathbf{r}_j(t) = \{r_{i,h}(t)\},\ r_{i,h}(t) \in \{0,1\},\ i \in \mathcal{I}_j,\ h \in \mathcal{H}$ represents the user request for tasks within the coverage of UAV j at the start of the tth time step;
- $\mathbf{N}_j(t-1) = \{n_{h,j}(t-1)\},\ h \in \mathcal{H}$ represents the count of instances of service h exists in UAV j at the conclusion of time step $t-1$, i.e., at the start of time step t;
- $\boldsymbol{\epsilon}_j(t) = \{\epsilon_{i,j}^{LoS}(t), \epsilon_{i,j}^{NLoS}(t)\},\ i \in \mathcal{I}_j$ represents the time-dependant additional path loss encountered during LoS and NLoS transmissions;
- $\mathbf{X}_j(t) = \{x_{i,j}(t)\},\ i \in \mathcal{I}_j$ represents the flat distance between UAV j and the UE i that falls within its coverage area;
- $\mathbf{Y}_j(t) = \{y_j(t)\}$ is the altitude of UAV j away from the ground.

Once the observation is obtained, each UAV will send it to the centre controller in the primary UAV, based on which the centre controller can obtain the joint state of of time slot t, which is expressed as

$$\mathbf{s}(t) \triangleq \{o_1(t),\ o_2(t), ..., o_J(t)\}, \quad (28)$$

and the state space could be represented by $\mathcal{S} = \{\mathbf{s}(t), t \in \mathcal{T}\}$.

2) Action Space: Under observation $o_j(t)$, the actor of agent j calculates its action $\mathbf{a}_j(t)$ as

$$\mathbf{a}_j(t) \triangleq \{\boldsymbol{\Delta}_j(t),\ \boldsymbol{\varrho}_j(t),\ \mathbf{A}_j(t),\ \mathbf{P}_j(t),\ \mathbf{F}_j(t)\}, \quad (29)$$

where

- $\boldsymbol{\Delta}_j(t) = \{\delta_{h,j}(t)\},\ h \in \mathcal{H}$ is the task placement decision at UAV j;
- $\boldsymbol{\varrho}_j(t) = \{\varrho_i(t)\},\ i \in \mathcal{I}_j$ is the offloading decision of UEs covered by UAV j;
- $\mathbf{A}_j(t) = \{a_{i,j}(t), a_{i,j,h}^{idle}(t), a_{i,j,h}^{new}(t)\},\ i \in \mathcal{I}_j,\ h \in \mathcal{H}$ represents the access control and instance selection;
- $\mathbf{P}_j(t) = \{p_{i,j}(t)\},\ i \in \mathcal{I}_j$ stands for the transmit power control of UEs served by UAV j;
- $\mathbf{F}_j(t) = \{f_{i,j}(t)\},\ i \in \mathcal{I}_j$ represents the computation resource allocation at UAV j.

The collection action of the J agents is described as

$$\mathbf{a}(t) = \{\mathbf{a}_1(t),\ \mathbf{a}_2(t), ..., \mathbf{a}_J(t)\}, \quad (30)$$

and accordingly, the action space could be expressed as $\mathcal{A} = \{\mathbf{a}(t), t \in \mathcal{T}\}$.

3) Reward Function: Upon taking action $\mathbf{a}_j(t)$, each agent j obtains an immediate reward $r(t)$. In our formulated problem $(\mathcal{P}_\mathbf{1})$, our objective is to reduce the long-term total cost incurred by all UEs. All agents collaborate to maximize their contributions towards minimizing the total cost, and each agent receives the same reward. In this context, we define the immediate reward $r(t)$ as the reciprocal of the total cost. Mathematically, it could be represented as

$$r(t) = r_j(t) = \frac{1}{J}\frac{1}{\sum\limits_{i\in\mathcal{I}} Cost_i(t)}. \quad (31)$$

Moreover, the actions of all agents are bounded by the constraints as defined by (C1)-(C15) in $(\mathcal{P}_\mathbf{1})$. However, the outputs of the actor networks may be infeasible due to these constraints. To address the issue, we need to judge whether the constraints are satisfied when all agents have taken their

actions. If the constraints are met, each agent will obtain its reward, and otherwise, it will receive a penalty, which is a dramatically small negative constant. By this manners, actors are guided to choose better actions in the next time slot. On the basis of the aforementioned analysis, the immediate reward for each agent j could be given by

$$r(t) = \begin{cases} \frac{1}{J}\frac{1}{\sum_{i\in\mathcal{I}} Cost_i(t)}, & \text{if all constraints are met,} \\ penalty, & \text{otherwise.} \end{cases} \tag{32}$$

4) Next State: After selecting and executing actions, each agent receives the immediate reward as defined above. Subsequently, the system transitions from the current state $\mathbf{s}(t)$ to the next state $\mathbf{s}(t+1)$ through the following means:

- User requests for tasks $r_{i,h}(t+1),\ i \in \mathcal{I},\ h \in \mathcal{H}$ transforms following a certain distribution, and in this paper, we consider it transforms according to uniform distribution;
- The count of service h's instances exists in UAV j at the end of time slot $t-1$, denoted as $n_{h,j}(t-1),\ j \in \mathcal{J},\ h \in \mathcal{H}$, undergoes changes based on the action $a_j(t)$ during the service placement stage. Specifically, the count at time slot t is given by $n_{h,j}(t) = n_{h,j}(t-1) + \delta_{h,j}(t)$, $j \in \mathcal{J},\ h \in \mathcal{H}$;
- Time-varying additional path loss under LoS and NLoS transmissions, i.e., $\{\epsilon_{i,j}^{LoS}(t+1), \epsilon_{i,j}^{NLoS}(t+1)\},\ i \in \mathcal{I},\ j \in \mathcal{J}$, are generated from the four tuples randomly: {0.1, 21}, {1, 20}, {1.6, 23}, and {2.3, 34};
- The horizontal distance between user i and UAV j $x_{i,j}(t+1) = 300\sin\theta_x + 50$, θ_x is randomly generated from $[0, \pi/2]$;
- Altitude of each UAV j away from the ground, i.e., $y_j(t+1) = 100\sin\theta_y + 50$, θ_y is randomly generated from $[0, \pi/2]$.

C. Reformulating Actions to Adapt to MADDPG

Since MADDPG is designed for addressing problems with continuous action spaces, each agent j will output continuous actions. To adapt the output of actors to our formulated problem $(\mathcal{P}_\mathbf{1})$, we need to transform the output of the actor networks.

1) Continuous Action Normalization: In order to cut down the action space's dimension, all continuous variables, including power control and computation resource assignment, are limited to a value between 0 and 1. Let a denote a general action output by the actor network, a^{max} and a^{min} denote the maximum and minimum value that can be taken by a, respectively, and let $\tilde{a}$ denote the actual value, then $\widetilde{a}$ can be recovered from a as follows:

$$\widetilde{a} = a(a^{max} - a^{min}) + a^{min}. \tag{33}$$

2) Discrete Action Reformulation: We transform the discrete actions as follows:

(i) For the integer task placement variable $\delta_{n,j}(t)$, we simply round it to the nearest integer, which is denoted as $\widetilde{\delta}_{n,j}(t)$.

(ii) For the binary offloading decision making $\varrho_i(t)$, we consider the output as the probability of taking $\varrho_i(t) = 1$. Note that, in order to balance the exploration-utilization issues in DRL, in the training process, the continuous output of $\varrho_i(t)$ is used as the probability for sampling the binary offloading decision, and in testing process, the agent will select 0 or 1 that with the maximum probability according to the value of $\varrho_i(t)$. We denote the reformulated offloading decision as $\widetilde{\varrho}_i(t)$.

(iii) For the binary variables including access control $a_{i,j}(t)$ and the instance selection variable $a_{i,j,h}^{idle}(t),\ a_{i,j,h}^{new}(t)$, since they are coupled by Eq. (6), i.e., constraint (C7) as in our formulated problem in $(\mathcal{P}_\mathbf{1})$, they should be addressed carefully.

We consider the output of $a_{i,j}(t)$ and $a_{i,j,h}^{idle}(t)$ as the probabilities of $a_{i,j}(t) = 1$ and $a_{i,j,h}^{idle}(t) = 1$. During the training process, a random value is taken following the probability to determine $a_{i,j}(t)$. If $a_{i,j}(t)$ is sampled as 0, then we take $a_{i,j,h}^{new}(t) = 0$ and $a_{i,j,h}^{idle}(t) = 0$. If $a_{i,j}(t)$ is sampled as 1, then we use a random probability to determine $a_{i,j,h}^{idle}(t)$: if $a_{i,j,h}^{idle}(t)$ is sampled as 1, we take $a_{i,j,h}^{new}(t) = 0$; otherwise, if $a_{i,j,h}^{idle}(t)$ is sampled as 0, we take $a_{i,j,h}^{new}(t) = 1$.

During the testing process, agent selects 0 and 1 that with the maximum probability for $a_{i,j}(t)$, if $a_{i,j}(t) = 0$, we will take $a_{i,j,h}^{new}(t) = 0$ and $a_{i,j,h}^{idle}(t) = 0$; if $a_{i,j}(t) = 1$, agent then selects 0 and 1 that with the maximum probability for $a_{i,j,h}^{idle}(t)$, and if $a_{i,j,h}^{idle}(t) = 1$, we take $a_{i,j,h}^{new}(t) = 0$, otherwise, if $a_{i,j,h}^{idle}(t) = 0$, and we take $a_{i,j,h}^{new}(t) = 1$.

Denote the reformulated access control and the instance selection variables as $\widetilde{a}_{i,j}(t)$ $\widetilde{a}_{i,j,h}^{idle}(t)$, and $\widetilde{a}_{i,j,h}^{new}(t)$, respectively, and the addressed action at time step t is expressed as

$$\widetilde{\mathbf{a}}_j(t) \triangleq \{\widetilde{\mathbf{\Delta}}(t),\ \widetilde{\boldsymbol{\varrho}}(t),\ \widetilde{\mathbf{A}}(t),\ \widetilde{\mathbf{P}}(t),\ \widetilde{\mathbf{F}}(t)\}. \tag{34}$$

Then the joint action of J agents can be denoted as $\widetilde{\mathbf{a}}(t) = \{\widetilde{\mathbf{a}}_1(t),\ \widetilde{\mathbf{a}}_2(t), ..., \widetilde{\mathbf{a}}_J(t)\}$.

D. Joint Optimization Algorithm Framework Based on MADDPG for our MINLP problem

Based on the aforementioned state space, action space, the addressed action, the reward, and the next state, the joint optimization algorithm based on MADDPG for MINLP is presented in Algorithm 1. For better understanding, Fig. 2 presents the framework of our Algorithm 1, which encompasses service placement, task offloading, access control, and resource allocation. As shown in Fig. 2 and Algorithm 1, at the beginning of each episode, each agent j observes the environment and acquires its initial state $o_j(t)$, and the central controller deployed at the primary UAV obtains the global state $\mathbf{s}(t)$. Based on $\mathbf{s}(t)$, each actor obtains its action $\mathbf{a}_j(t) = \mu_j(\mathbf{s}(t); \theta_j)$, i.e., the joint task placement, offloading decision, access control and instance selection, transmit power control, and computation resource assignment, by providing the state as input to the actor network. In order to enhance the exploration capabilities of the algorithm, a noise $\Psi(t)$ which is randomly sampled from an Ornstein-Uhlenbeck process is incorporated into $\mu_j(\mathbf{s}(t); \theta_j)$, and the action becomes $\mathbf{a}_j(t) = \mu_j(\mathbf{s}(t); \theta_j) + \Psi(t)$. Then transform action as in Section V-C, $\mathbf{a}_j(t)$ becomes valid action $\widetilde{\mathbf{a}}_j(t)$. Each agent executes its corresponding action $\widetilde{\mathbf{a}}_j(t)$, and it will receive its reward $r_j(t)$, and its observation will transit to $o_j(t+1)$, and

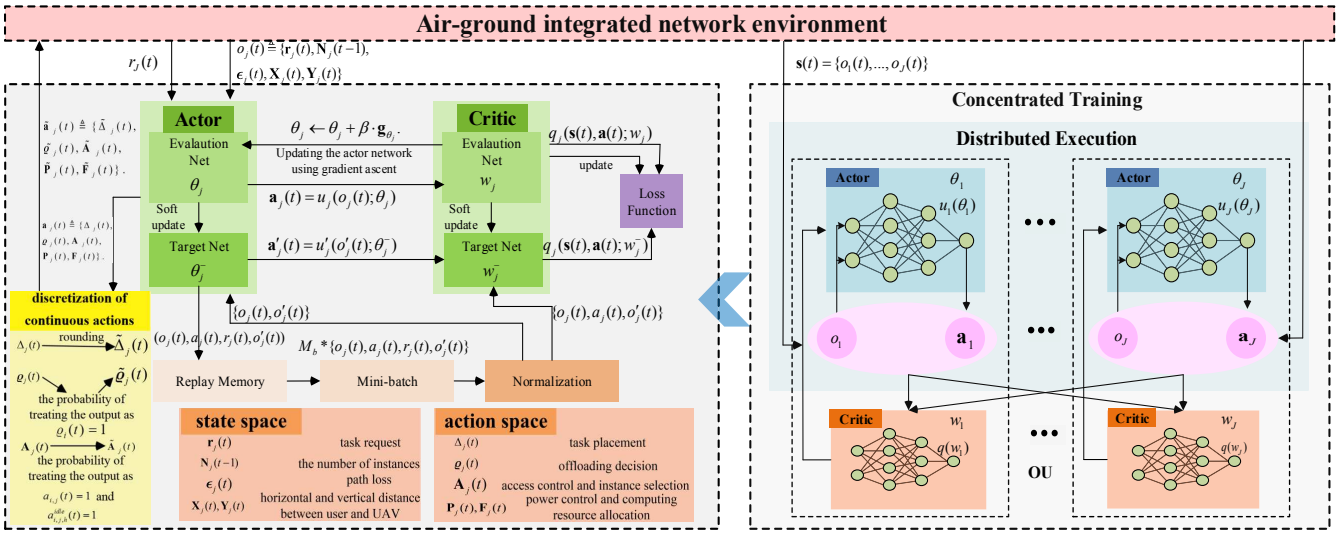


Fig. 2: Framework of the proposed MINLP Tailored MADDPG algorithm.

meanwhile, the global environment will transform to the next state $\mathbf{s}(t+1)$.

Then we store the experience $<\mathbf{s}(t), \mathbf{a}(t), r(t), \mathbf{s}(t+1)>$ into the experience pool. If the experience pool becomes full, randomly choose M samples from it to create a mini-batch. This mini-batch will be utilized to train the actor and critic networks, according to the procedures described in equations (18) and (24). Finally, all target networks are updated according to equation (25).

E. Complexity Analysis

In each training step, the MADDPG algorithm samples and stores experiences from the J agents, and then sample batch of M experiences from the stored memory for training each agent. During each time step, the MADDPG algorithm computes and updates the action policies and value functions of each agent, which involves the forward and backward propagation of neural networks. The MADDPG algorithm utilizes deep neural networks to construct the actor and critic networks, which consist of input layers, hidden layers, and output layers. Specifically, the actor network takes the state space as input and produces actions as output, while the critic network takes the concatenation of the state and action spaces as input and outputs Q-values. The primary factor influencing the time complexity is the dimensionality of the network structures. The computational complexity required for training each actor using a piece of experience is $O_{actor} = O(RH + H^2 + HL)$, where R represents the dimensionality of the state space, H represents the number of neurons in the hidden layers (there are 2 hidden layers in our network), and L represents the dimensionality of the action space. Similarly, the time complexity of the critic network can be represented as $O_{critic} = O((R + L)H + H^2 + H)$. Since the target actor and critic networks have the same network structure as the actor and critic networks, the algorithm complexity for a single agent can be expressed as follows $O_{single} = O\Big(2(RH+H^2+HL)+2((R+L)H+H^2+H)\Big)$. Therefore, we can determine the overall time complexity of the algorithm is $O = O\Big(2JM(RH+H^2+HL+(R+L)H+H^2+H)\Big)$.

VI. Simulation Results and Discussions

In this section, we assess the performance of our proposed MADDPG algorithm for MINLP through simulation. Default parameter settings are exhibited in Table I, and they will keep unchanged in the following simulations unless otherwise stated. Our simulation was run on python 3.7, pycharm 2021 and tensorflow 1.14.

In the subsequent analysis, we initially assess the convergence of the proposed MADDPG algorithm for MINLP under various learning rates. Subsequently, we compare its performance with several baseline algorithms to provide a comprehensive evaluation. To simplify notation, we refer to our proposed MINLP tailored MADDPG algorithm as "Proposed" or "our proposed algorithm" in the subsequent discussions, which means that we conduct the joint optimization as defined in problem $(\mathcal{P}_1)$. The comparison algorithm are given as follows.

1) "POPO (placement-offloading-power-optimization)": the access control strategy takes random values and other strategies are optimized using our proposed algorithm. POPO shares similarities with the approach presented in [39], where optimizations are performed for task placement, offloading decisions, and transmit power control. Besides, POPO also optimize computation resource allocation in UAVs. Furthermore, while each service in [39] can serve all tasks of the service, in POPO, each service instance can only serve a single task.

2) "OTPOA (Only-task-placement-offloading-access)": only the task placement, task offloading, and access control policies are optimized using our proposed algorithm, while the transmit power control and computing resource allocation policies are randomized.

3) "All-offloading": all UEs perform task offloading, i.e., the offloading decision of all UEs equals to 1, and other policies, including task placement, access control, transmission power

TABLE I: Parameter Notations and Default Value Settings

Parameter	Notations	Value
Number of UEs	I	10 [37]
Number of UAVs	J	3 [14]
Number of tasks	H	10
UEs' local processing capability	f_i^l	0.06 G CPU cycles/s [9]
UEs' maximum transmit power	p^{max}	0.1 W [16]
Input data size of task	D_h	8 Mb [7]
Processing density of task	Ξ_h	1000 CPU cycles/s
Size of each task instance	w_h	800 Mb
Delay tolerance of tasks	T_h^{max}	1 s [14]
Processing capability of MEC server	f_j^{max}	12 G CPU cycles/s [16] [6]
Storage capacity of each MEC server	Π_j	3 G
Bandwidth of each UAV	B_j	(1,5) GHz [23]
Carrier frequency	f_c	1.2×10^9 Hz
Speed of light	c	3×10^8 m/s [23]
Environment-determined variables	α_1 and α_2	1.88, 0.43 [38]
The power spectral density of noise	σ^2	17×10^{-12} [18]
Price of UAV communication resource	$\eta^{j,1}$	1×10^{-9} \$/bps
Price of UAV computation resource	$\eta^{j,2}$	4×10^{-9} \$/(CPU cycles/s)
Price of local computation resource	η_i^l	0.2×10^{-9} \$/(CPU cycles/s)
Service instance initializing delay	$D_{h,j}^{place}$	0.1 s
Cost for using a new deployed instance	$\lambda_{h,j}$	0.001 \$
Weight of delay	γ^1	0.3 Utility/s
Weight of expense	γ^2	0.7 Utility/\$
Cost of delay	ϑ^t	0.008 Cost/s
Cost of expense	ϑ^e	0.004 Cost/\$
Task placement threshold	$thres$	3
Access distance threshold	D_{th}	300 m [3]

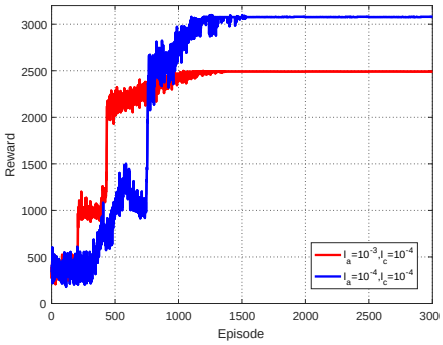


Fig. 3: Convergence of our proposed algorithm under various learning rate for the actor network.

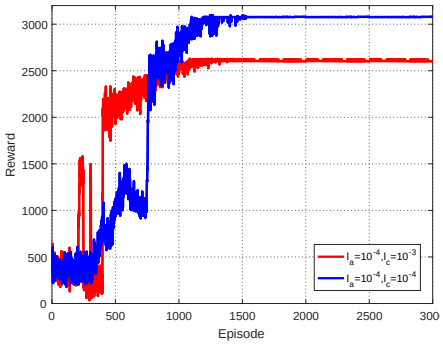


Fig. 4: Convergence of our proposed algorithm under various learning rate for the critic network.

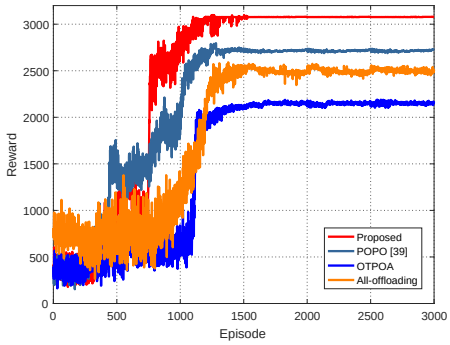


Fig. 5: Comprehensive performance comparison between the four algorithms.

control, and computing resource allocation are optimized using our proposed algorithm.

4) "Random": all decisions are made randomly, and there's no concepts of states, actions and training process as in other algorithms. Specifically, in this algorithm, the states variables as in the proposed algorithm, will act as general known quantities, and these quantities are coupled between different time slots as the state transition of our proposed algorithm. Also, the optimization variables, i.e., the actions as in Proposed, are also considered as known quantities, which are generated randomly from their value range.

A. Convergence of Algorithm 1

Figs. 3 and 4 demonstrate the convergence of our proposed algorithm under different combinations of learning rates. The x-axis of both figures stands for the episodes, while the y-axis means the immediate total reward of all agents in each episode. Notably, it is evident from both figures that our proposed algorithm achieves convergence regardless of the selected learning rate. In Fig. 3, we set the learning rate of the critic network as $l_c = 10^{-4}$, and for the actor network, we evaluate two different learning rates, namely $l_a = 10^{-3}$ and $l_a = 10^{-4}$. In Fig. 4, we maintain the learning rate of the actor network at $l_a = 10^{-4}$, while examining the impact of different learning rates for the critic network, specifically $l_c = 10^{-3}$ and $l_c = 10^{-4}$. Both the curves in the two figures converge at nearly the 1500th episode, which proves that our algorithm converges quickly in solving our complicated problem $(\mathcal{P}_1)$.

Fig. 5 illustrates the performance and convergence of the above mentioned four algorithms, i.e., Proposed, POPO, OT-

Algorithm 1 Joint Optimization Algorithm Based on MADDPG for MINLP

Initialization:

Initialize the weight parameters θ_j and w_j of the primary actor and critic, and initialize the parameters θ_j^- and w_j^- for the target actor and critic by letting $\theta_j \leftarrow \theta_j^-$ and $\alpha_j \leftarrow \alpha_j^-$, respectively.
Initialize the learning rate α and β corresponding to the critic and actor networks, the discount factor γ, the number of episode EP, and the maximum number of training steps per episode T^{max}.
Initialize the experience pool D, the size of mini-batch M, and the random process Ψ for action exploration.
Initialize the network layout parameters, such as the number of UEs I, the number of UAVs J, the parameters of tasks, etc.

Iteration:

for episode =1:EP **do**
  Each UAV is initialized with an initial state $o_j(t)$, and the central controller obtains the global state $\mathbf{s}(t)$.
  for each step $t = 1 : T^{max}$ **do**
    for agent $j = 1 : J$ **do**
      Takes action $\mathbf{a}_j(t) = \mu_j(\mathbf{s}(t); \theta_j) + \Psi(t)$.
      Reform action $\mathbf{a}_j(t)$ as $\widetilde{\mathbf{a}}_j(t)$.
      Execute the action $\widetilde{\mathbf{a}}_j(t)$, receives its reward $r(t)$, and the next observe $o_j(t+1)$.
    end for
    Obtain $\mathbf{s}(t)$ and $\mathbf{a}(t)$ according to (28) and (30).
    Store $< \mathbf{s}(t), \tilde{\mathbf{a}}(t), \mathbf{r}(t), \mathbf{s}(t+1) >$ in experience pool D.
    Set $\mathbf{s}(t) \leftarrow \mathbf{s}(t+1)$.
    if Reply buffer is full **then**
      for agent $n = 1 : J$ **do**
        Randomly select M samples from the experience pool to create a mini-batch.
        Update the primary actor and critic networks via (18) and (24), respectively.
        Update the target networks via (25).
      end for
    end if
  end for
end for

POA, and All-offloading. In the comparison, all parameters take their default settings. From the figure, it can be observed that all four algorithms exhibit good convergence performance. However, among these algorithms, our proposed algorithm is superior to the other three in both convergence speed and reward performance. In convergence speed, the proposed algorithm converges slightly faster than other algorithms, and when it converges, the reward almost keeps stable in the following episodes, while the baseline algorithms still fluctuate to a certain extent after convergence.

Next we compare the reward performance among Proposed and the baseline algorithms. Since our objective is to minimize the total cost of the system, while the reward is defined as the reciprocal of the total cost, it follows that a higher reward corresponds to better performance. Thus, in the subsequent simulations, higher reward values indicate improved performance in terms of cost reduction. Among the baseline algorithms, POPO does not incorporate an optimized access control strategy, so the performance gap between it and our proposed algorithm can therefore reflect the performance improvement achieved through access control optimization. Similarly, OTPOA does not optimize the transmit power and computation resource allocation strategies, and thus, the performance gap between OTPOA and our proposed algorithm highlights the performance gain resulting from optimized transmit power control; and All-offloading does not incorporate an optimized offloading strategy, and therefore, the performance gap observed between it and our proposed algorithm can be attributed to the performance improvement resulting from optimized task offloading decision-making. Based on the performance depicted in Fig. 5, it is evident that our proposed algorithm achieves the best performance among the evaluated algorithms. It is followed by POPO, All-offloading, and finally OTPOA, in descending order of performance.

In Fig. 6, we plot the curve depicting the relationship between local processing capacity of UEs and the average system reward of each episode after the algorithm converges (which is shorted as system reward). When the local processing capacity increases, except for All-offloading, the rewards of other algorithms gradually increase. This is because when local processing capacity increase, task processing delay will be reduced, and the cost will be reduced for one hand; moreover, for Proposed, POPO, and OTPOA, when local processing capacity increases, local processing will be feasible for many UEs, and the three algorithms will let more UEs to process their tasks locally by task offloading optimization. As a result, little economic expenditure will be spend, and the cost will further be reduced for another hand. When cost is reduced, the system immediate reward will be increased. For All-offloading algorithm, since all UEs offload their tasks to UAV processing, the local processing capability of UEs will not affect the system cost, and further has no effect on the system reward it could obtain. However, since All-offloading algorithm offloads all tasks to the UAV for execution, the local processing capability of UEs will not affect the system cost and the system reward it could obtain, making the reward of the All-offloading algorithm remains relatively constant as the local processing capacity increases. From Fig. 6, it is evident that our proposed algorithm consistently maintains the highest reward. This demonstrates the robustness and effectiveness of our algorithm in achieving optimal performance regardless of changes in local processing capacity.

In Fig. 7, we present the relationship between the input data size D_h and the system reward. As was defined, the computation amount is defined as $C_h = D_h \Xi_h$, which is tightly coupled with the input data size D_h. When the task data size is very small, the computation amount C_h is also very tiny, the majority of tasks can be effectively executed locally. As a result, the rewards achieved by the Proposed, POPO, and OTPOA are very similar. In this scenario, the need for task offloading is minimal, and the impact of factors related to edge processing, such as access control and power control strategies, is relatively insignificant. Consequently, the performance of these algorithms is very close, leading to similar reward values. For All-offloading algorithm, all tasks are offloaded to the UAV for processing, which increases the processing latency and incurs higher economic expenditure, leading to a decrease in the obtained reward. As the input data size D_h increases, the rewards of all four algorithms gradually decrease. This is because the data transmit delay and

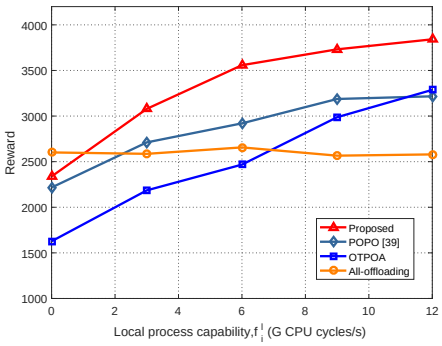


Fig. 6: Reward vs. local processing capability of UEs f_i^l.

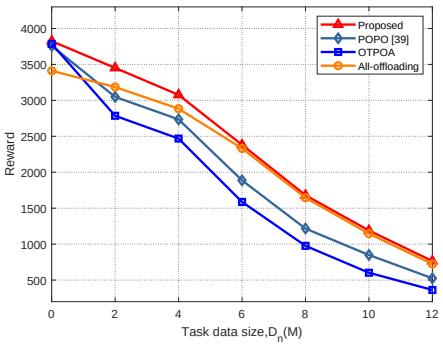


Fig. 7: Reward vs. input data size D_h.

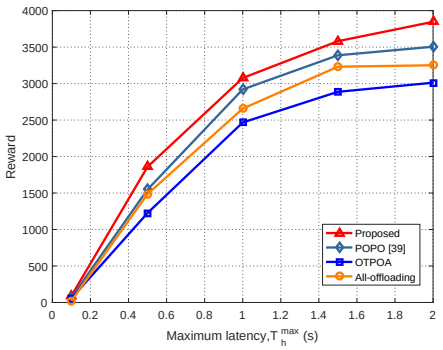


Fig. 8: Reward vs. the maximum task processing latency T_h^{max}.

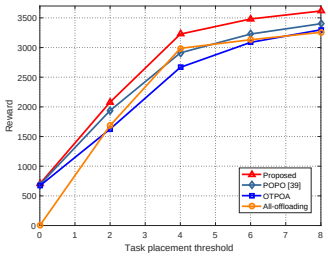


Fig. 9: Reward vs. task placement threshold $thre$.

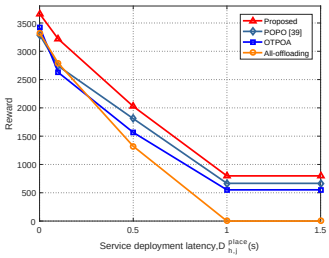


Fig. 10: Reward vs. service deployment latency $D_{h,j}^{place}$.

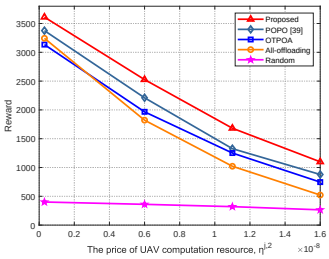


Fig. 11: Reward vs. UAVs' computation resource price $\eta^{j,2}$.

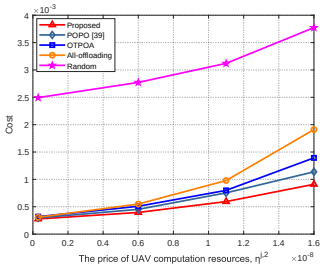


Fig. 12: Cost vs. UAVs' computation resource price $\eta^{j,2}$.

the fees for data transmission in task offloading will increase, which will contribute to a decrease in the system reward. When D_h is greater than 6 Mbit, since the default processing density is defined as $\Xi_h = 1000$ in this paper, the computation amount C_h will be quite large, and at this point, the optimal choice will be offloading all tasks to the edge servers for processing, which is achieved by All-offloading. Because joint access control, task placement and task offloading decision optimization, our Proposed algorithm could achieve nearly the same performance with All-offloading, which reflects in Fig. 7 that they both could obtain nearly the same reward.

In Fig. 8, we present the relationship between the maximum task processing tolerable latency T_h^{max} and the system reward. When T_h^{max} is extremely low, indicating stringent latency requirements, significant computational resources are required for task processing. At this stage, none of the four algorithms can handle tasks locally, and task offloading becomes necessary. However, the data transmission in task offloading, service deployment, and task processing by the UAV all introduce certain latency, which cannot be completed within the limited time of T_h^{max}, and UAV processing will also infeasible. Consider the reward function that we defined earlier, when any constraints could not be satisfied, the reward will be zero. Therefore, the obtained reward of all algorithms are zero at this point.

As the maximum tolerable latency for task processing increases to 0.5s, it becomes feasible to process some tasks either locally or on the UAV. Therefore, the system's reward grows rapidly at this stage for all algorithms, and this increase continues until $T_h^{max} = 1$ seconds. When $T_h^{max} > 1$, most tasks can be processed by the system, so the reward growth slows down. As $T_h^{max} > 1.5$, nearly all tasks can be successfully executed by a UAV or the UE itself. As a result, the reward of All-offloading algorithm stops growing, because there is no further performance gain from task offloading optimization. For the other three algorithms, as they have the flexibility to choose between local and UAV execution based on optimization criteria, they can effectively decrease latency and economic costs, resulting in a slight increase in system reward. What's more, when the maximum task processing latency increases, our proposed algorithm consistently outperforms All-offloading, POPO, and OTPOA in terms of reducing overall system cost and therefore maximizing the system reward.

In Fig. 9, we depict the relationship between the task placement threshold $thre$ and the system reward. When $thre = 0$, it implies that the UAVs are unable to deploy and handle any tasks. In such a scenario, our proposed algorithm, POPO, and OTPOA utilize task offloading decision optimization to determine that tasks should be processed locally, thereby enabling them to receive a certain amount of reward. However, in most cases, the limited processing capability of UEs often results in the inability to accomplish a significant portion of tasks, making the QoS constraints are frequently not met, and thus the reward is usually 0 as defined by our reward function in Eq. (31). Consequently, the system reward is generally low in such scenarios. For All-offloading, tasks can only be offloaded to UAVs for processing. However, since the UAVs are incapable of handling any tasks, All-offloading will receive a reward of 0 in this particular case. As $thre$ increases from 0 to 4, the reward for all four algorithms experiences a significant growth. This can be attributed to the ability to deploy more instances

at UAVs, resulting in an increased capacity for processing tasks with lower latency, and therefore more reward. As $thre$ continues to increase, the reward for the four curves grows at a slower pace. In this scenario, even though each UAV can be deployed with more instances, the actual number of instances that can be deployed is constrained by the storge capability of UAVs. As a result, the incremental benefits in reward derived from increasing $thre$ become progressively smaller over time. Moreover, it is noteworthy that our proposed algorithm consistently demonstrates the highest reward when the task placement threshold is fixed at any given point.

In Fi. 10, we illustrate the correlation between the system reward and the service deployment latency $D_{h,j}^{place}$. As indicated in Eq. (7), the task placement solely influences the case of selecting a newly initialized instance in the UAV processing mode. When $D_{h,j}^{place}$ is extremely small, the task completion delay is primarily attributed to data transmission and processing, with no additional time spent on service instance placement. As a result, the reward is significantly higher. As $D_{h,j}^{place}$ increases, the data transmission and processing may exceed the available time frame of $T_h^{max} - D_{h,j}^{place}$ for numerous UEs when utilizing a newly initialized service instance. Consequently, the system reward experiences a rapid decline. When $D_{h,j}^{place} > 1$, i.e., the service placement delay exceeds the maximum required task completion delay, the QoS requirement (C15) cannot be fulfilled when utilizing a newly initialized service instance. Consequently, the reward in this scenario will be zero. Since local processing and using an existing instance for task processing may be feasible, our proposed algorithms, POPO and OTPOA, can still yield certain system rewards by leveraging such situations. For All-offloading, the service deployment latency is extremely high and the storage capacity of UAV edge nodes is limited, tasks cannot be completed within T_h^{max}. Consequently, the reward in these cases will be zero. Furthermore, it is evident that our proposed algorithm consistently outperforms others regardless of the values of $D_{h,j}^{place}$.

In Figs. 11 and 12, we demonstrate the system reward and cost versus the price of UAVs' computation resources $\eta^{j,2}$. There are two points that should be noted. (i) In the two figures, we add Random as a baseline algorithm. (ii) The cost in Fig. 12 equals to the reciprocal of the reward of each episode, which is motivated from the definition of reward as in (32). However, (32) is defined for each time slot, and the cost in Fig. 12 is defined for each episode. Although the time scale is not exactly the same, the cost in Fig. 12 can also reflect the cost of the system.

When the price of UAVs' computation resource is very low, except for Random, all other algorithms could gain relatively large reward, and correspondingly, the cost is very small. With the increase of $\eta^{j,2}$, more money will be spent in task processing, and the reward of all algorithms decrease, and the costs increase meanwhile. Since there's no any optimization in Random, the performance of Random is very poor, where the reward of Random is much smaller and the cost of Random is much higher than other algorithms as shown in Figs. 11 and 12. It can also be observed that, the proposed algorithms always performs the best under different values of $\eta^{j,2}$.

VII. Conclusions

In this paper, we have conducted a comprehensive investigation of the task offloading process in an AGIN, encompassing both the service placement and task offloading stages. In service deployment stage, we optimized the application placement and replacement issues, with the constraints on the storage capacity satisfied. In the task offloading stage, we focused on optimizing several key aspects, including the task offloading decision, access control and service instance selection, transmit power control for each UE, and computation resource assignment for each UAV. Since the problem is a MINLP problem with tightly coupled optimization variables, it poses significant challenges for traditional optimization techniques to find an optimal solution efficiently. Also, as we intended to realize long-term system cost minimization, and our problem contains continuous variables, which motivates us to use MADDPG to solve the problem. We further improve MADDPG to adapt to our MINLP problem in solving the integer variables with the coupling between variables guaranteed. Simulation results have demonstrated the efficacy of our proposed algorithm, which exhibits fast convergence and outperforms other baseline algorithms in terms of maximizing the reward, i.e., weighted sum of task processing delay and economic expenditure cost reduction.

References

[1] D. Zhou, M. Sheng, J. Wu, J. Li, and Z. Han, "Gateway placement in integrated satellite-terrestrial networks: Supporting communications and internet of remote things," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4421–4434, 2022.

[2] D. Zhou, M. Sheng, Y. Wang, J. Li, and Z. Han, "Machine learning based resource allocation in satellite networks supporting internet of remote things," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6606–6621, 2021.

[3] Z. Zhang, Q. Zhang, J. Miao, F. R. Yu, F. Fu, J. Du, and T. Wu, "Energy-efficient secure video streaming in uav-enabled wireless networks: A safe-dqn approach," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 4, pp. 1892–1905, 2021.

[4] J. Liu, X. Zhang, R. Zhang, T. Huang, and F. R. Yu, "Reliable and low-overhead clustering in leo small satellite networks," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14 844 – 14 856, 2022.

[5] J. Kang, H. Du, Z. Li, Z. Xiong, S. Ma, D. Niyato, and Y. Li, "Personalized saliency in task-oriented semantic communications: Image transmission and performance analysis," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2022.

[6] X. Ren, C. Qiu, X. Wang, Z. Han, K. Xu, H. Yao, and S. Guo, "Ai-bazaar: A cloud-edge computing power trading framework for ubiquitous ai services," *IEEE Transactions on Cloud Computing*, vol. early access, no. doi: 10.1109/TCC.2022.3201544, 2022.

[7] L. Liu, J. Feng, X. Mu, Q. Pei, D. Lan, and M. Xiao, "Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. early access, no. doi: 10.1109/TITS.2023.3249745, 2023.

[8] X. Yuan, J. Chen, N. Zhang, J. Ni, F. R. Yu, and V. C. Leung, "Digital twin-driven vehicular task offloading and irs configuration in the internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24 290–24 304, 2022.

[9] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, "Mec-assisted immersive vr video streaming over terahertz wireless networks: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9517–9529, 2020.

[10] X. Yuan, J. Chen, J. Yang, N. Zhang, T. Yang, T. Han, and A. Taherkordi, "Fedstn: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. early access, 2022.

[11] R. Zhang, J. Liu, F. Liu, T. Huang, Q. Tang, S. Wang, and F. R. Yu, "Buffer-aware virtual reality video streaming with personalized and private viewport prediction," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 2, pp. 694–709, 2021.

[12] S. Bi, L. Huang, and Y. J. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4947–4963, 2020.

[13] Z. Cheng, M. Min, M. Liwang, L. Huang, and Z. Gao, "Multiagent ddpg-based joint task partitioning and power control in fog computing networks," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 104–116, 2021.

[14] S. Mao, S. He, and J. Wu, "Joint uav position optimization and resource scheduling in space-air-ground integrated networks with mixed cloud-edge computing," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3992–4002, 2020.

[15] H. Guo, X. Zhou, Y. Wang, and J. Liu, "Achieve load balancing in multi-uav edge computing iot networks: A dynamic entry and exit mechanism," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18 725–18 736, 2022.

[16] G. Zheng, C. Xu, M. Wen, and X. Zhao, "Service caching based aerial cooperative computing and resource allocation in multi-uav enabled mec systems," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 10, pp. 10 934–10 947, 2022.

[17] X. Duan, Y. Zhou, D. Tian, J. Zhou, Z. Sheng, and X. Shen, "Weighted energy-efficiency maximization for a uav-assisted multiplatoon mobile-edge computing system," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18 208–18 220, 2022.

[18] X. He, R. Jin, and H. Dai, "Multi-hop task offloading with on-the-fly computation for multi-uav remote edge computing," *IEEE Transactions on Communications*, vol. 70, no. 2, pp. 1332–1344, 2021.

[19] J. Kang, X. Li, J. Nie, Y. Liu, M. Xu, Z. Xiong, D. Niyato, and Q. Yan, "Communication-efficient and cross-chain empowered federated learning for artificial intelligence of things," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2022.

[20] M. Li, P. Si, R. Yang, Z. Wang, and Y. Zhang, "Uav-assisted data transmission in blockchain-enabled m2m communications with mobile edge computing," *IEEE Network*, vol. 34, no. 6, pp. 242–249, 2020.

[21] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/aerial-assisted computing offloading for iot applications: A learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1117–1129, 2019.

[22] H. Liao, Z. Wang, Z. Zhou, Y. Wang, H. Zhang, S. Mumtaz, and M. Guizani, "Blockchain and semi-distributed learning-based secure and low-latency computation offloading in space-air-ground-integrated power iot," *IEEE Journal of Selected Topics in Signal Processing*, vol. early access, no. 16, pp. 381–394, 2021.

[23] Y. Gong, H. Yao, D. Wu, W. Yuan, T. Dong, and F. R. Yu, "Computation offloading for rechargeable users in space-air-ground networks," *IEEE Transactions on Vehicular Technology*, vol. early access, 2022.

[24] H. Peng and X. Shen, "Multi-agent reinforcement learning based resource management in mec-and uav-assisted vehicular networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 131–141, 2020.

[25] Y. Chen, Y. Sun, B. Yang, and T. Taleb, "Joint caching and computing service placement for edge-enabled iot based on deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 19 501–19 514, 2022.

[26] X. Gao, R. Liu, and A. Kaushik, "Virtual network function placement in satellite edge computing with a potential game approach," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1243–1259, 2022.

[27] S. Yu, X. Gong, Q. Shi, X. Wang, and X. Chen, "Ec-sagins: Edge-computing-enhanced spacecaircground-integrated networks for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5742–5754, 2021.

[28] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5g ultradense network," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2238–2251, 2020.

[29] J. Du, Y. Sun, N. Zhang, Z. Xiong, A. Sun, and Z. Ding, "Cost-effective task offloading in noma-enabled vehicular mobile edge computing," *IEEE Systems Journal*, 2022.

[30] Y. Li, W. Liang, and J. Li, "Profit maximization for service placement and request assignment in edge computing via deep reinforcement learning," in *Proceedings of the 24th International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Alicante, Spain, Nov. 2021, pp. 51–55.

[31] H. Zhu, Q. Wu, X. Wu, J. Fan, P. Fan, and J. Wang, "Decentralized power allocation for mimo-noma vehicular edge computing based on deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 770–12 782, 2021.

[32] F. Fu, Y. Kang, Z. Zhang, F. R. Yu, and T. Wu, "Soft actorccritic drl for live transcoding and streaming in vehicular fog-computing-enabled iov," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1308–1321, 2021.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[34] T. P. Lillicrap, J. J. Hunt, A. Pritzel, and etc., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[35] J. Du, W. Cheng, G. Lu, H. Cao, X. Chu, and Z. Zhang, "Resource pricing and allocation in mec enabled blockchain systems: An a3c deep reinforcement learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 33 – 44, 2022.

[36] M. Li, F. R. Yu, P. Si, W. Wu, and Y. Zhang, "Resource optimization for delay-tolerant data in blockchain-enabled iot with edge computing: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9399–9412, 2020.

[37] K. Wei, Q. Tang, J. Guo, M. Zeng, Z. Fei, and Q. Cui, "Resource scheduling and offloading strategy based on leo satellite edge computing," in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*. Norman, OK, USA, Sep. 2021, pp. 1–6.

[38] R. I. Bor-Yaliniz, A. El-Keyi, and H. Yanikomeroglu, "Efficient 3-d placement of an aerial base station in next generation cellular networks," in *2016 IEEE international conference on communications (ICC)*. Kuala Lumpur, Malaysia, May. 2016, pp. 1–5.

[39] Z. Yao, Y. Li, S. Xia, and G. Wu, "Attention cooperative task offloading and service caching in edge computing," in *2022 IEEE Global Communications Conference*. Rio de Janeiro, Brazil, Dec. 5189-5194, pp. 1–5.

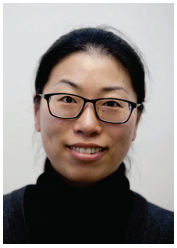
Jianbo Du received her Ph.D. in communication and information systems from Xidian University, Xi'an, Shaanxi, China, in 2018. She was a visiting scholar at Carleton University, Canada, in 2019. She is now an associate professor with the School of Communication and Information Engineering, Xi'an University of Posts and Telecommunications. Her research interests include mobile edge computing, blockchain, space-air-ground integrated networks, deep reinforcement learning, optimization theory and methods, etc., and their applications in wireless communications.

Kong Ziwen obtained a Bachelor's degree in Communication Engineering from Xi'an University of Technology in 2021. She is currently studying for a Master of Engineering degree in Information and Communication at Xi'an University of Posts and Telecommunications. Her research interests include mobile edge computing, blockchain and deep reinforcement learning algorithms, as well as their applications in wireless communication.

Aijing Sun is a professor and the Dean of the School of Communication and Information Engineering, Xi'an University of Posts and Telecommunications. She is the vice chairman of Shaanxi Institute of Communications, expert member of Communications Branch of China Institute of Electronics, etc. As the first complete person, she won the third prize of Shaanxi Science and Technology Award, the second prize of Shaanxi Provincial Higher Education Science and Technology Award, the first prize of Shaanxi Province Excellent Textbook, and the second prize of Shaanxi Provincial Higher Education Teaching Achievement Award. She is also the winner of the 2020 Shaanxi Provincial Teaching Teacher Award due to her excellent contributions in education and research. Her research interests include Internet of Things technology and intelligent education information technology.

Jiawen Kang received the Ph.D. degree from the Guangdong University of Technology, China in 2018. He has been a postdoc at Nanyang Technological University, Singapore from 2018 to 2021. He currently is a full professor at Guangdong University of Technology, China. His research interests mainly focus on blockchain, security, and privacy protection in wireless communications and networking.

Dusit Niyato (M'09-SM'15-F'17) is a professor in the School of Computer Science and Engineering, at Nanyang Technological University, Singapore. He received B.Eng. from King Mongkuts Institute of Technology Ladkrabang (KMITL), Thailand in 1999 and Ph.D. in Electrical and Computer Engineering from the University of Manitoba, Canada in 2008. His research interests are in the areas of sustainability, edge intelligence, decentralized machine learning, and incentive mechanism design.



Xiaoli Chu (M'06-SM'15) is a Professor in the Department of Electronic and Electrical Engineering at the University of Sheffield, UK. She received the B.Eng. degree in Electronic and Information Engineering from Xi'an Jiao Tong University in 2001 and the Ph.D. degree in Electrical and Electronic Engineering from the Hong Kong University of Science and Technology, Hong Kong, China, in 2005. From 2005 to 2012, she was with the Centre for Telecommunications Research at King's College London. Xiaoli has co-authored over 200 peerreviewed journal and conference papers, including 8 ESI Highly Cited Papers and the IEEE Communications Society 2017 Young Author Best Paper. She co-authored/co-edited the books FogEnabled Intelligent IoT Systems (Springer 2020), Ultra Dense Networks for 5G and Beyond (Wiley 2019), Heterogeneous Cellular Networks-Theory, Simulation and Deployment (Cambridge University Press 2013), and 4G Femtocells: Resource Allocation and Interference Management (Springer 2013). She is Senior Editor for IEEE Wireless Communications Letters, Associate Editor for IEEE Transactions on Network Science and Engineering, Editor for IEEE Open Journal of Vehicular Technology, and received the IEEE Communications Letters Exemplary Editor Award in 2018.

F. Richard Yu (S'00-M'04-SM'08-F'18) F. Richard Yu (Fellow, IEEE) received the Ph.D. degree in electrical engineering from The University of British Columbia (UBC) in 2003. From 2002 to 2006, he was with Ericsson, Lund, Sweden, and a start-up in California, USA. He joined Carleton University in 2007, where he is currently a Professor. His research interests include cross-layer/cross-system design, security, green ICT, and QoS provisioning in wireless-based systems. He received the IEEE Outstanding Service Award in 2016, the IEEE Outstanding Leadership Award in 2013, the Carleton Research Achievement Award in 2012, the Ontario Early Researcher Award (formerly Premiers Research Excellence Award) in 2011, the Excellent Contribution Award at IEEE/IFIP TrustCom 2010, the Leadership Opportunity Fund Award from Canada Foundation of Innovation in 2009, and the Best Paper Awards at IEEE ICC 2014, Globecom 2012, IEEE/IFIP TrustCom 2009, and Intl Conference on Networking 2005. He has served as the technical program committee (TPC) co-chair for numerous conferences. He is a registered Professional Engineer in the province of Ontario, Canada. He serves as the Vice-Chair for the IEEE Technical Committee on Green Communications and Computing and a member of Board of Governors for the IEEE Vehicular Technology Society.