Space/Aerial-Assisted Computing Offloading for IoT Applications: A Learning-Based Approach

Nan Cheng, *Member, IEEE*, Feng Lyu, *Member, IEEE*, Wei Quan, *Member, IEEE*, Conghao Zhou, *Senior Member, IEEE*, Hongli He, *Student Member, IEEE*, Weisen Shi, *Senior Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*— Internet of Things (IoT) computing offloading is a challenging issue, especially in remote areas where common edge/cloud infrastructure is unavailable. In this paper, we present a space-air-ground integrated network (SAGIN) edge/cloud computing architecture for offloading the computation-intensive applications considering remote energy and computation constraints, where flying unmanned aerial vehicles (UAVs) provide near-user edge computing and satellites provide access to the cloud computing. First, for UAV edge servers, we propose a joint resource allocation and task scheduling approach to efficiently allocate the computing resources to virtual machines (VMs) and schedule the offloaded tasks. Second, we investigate the computing offloading problem in SAGIN and propose a learning-based approach to learn the optimal offloading policy from the dynamic SAGIN environments. Specifically, we formulate the offloading decision making as a Markov decision process where the system state considers the network dynamics. To cope with the system dynamics and complexity, we propose a deep reinforcement learning-based computing offloading approach to learn the optimal offloading policy on-the-fly, where we adopt the policy gradient method to handle the large action space and actor-critic method to accelerate the learning process. Simulation results show that the proposed edge VM allocation and task scheduling approach can achieve near-optimal performance with very low complexity and the proposed learning-based computing offloading algorithm not only converges fast but also achieves a lower total cost compared with other offloading approaches.**

***Index Terms*— Computing offloading, edge computing, space-air-ground, IoT, reinforcement learning.**

I. Introduction

With the rapid development of 5G networks and Internet of things (IoT), a myriad of promising applications and services have emerged, such as virtual reality, HD live streaming, autonomous driving, industry automation, smart home, and so forth, which reap the benefits provided by 5G networks, such as ultra-high data rate, low latency, high reliability, and massive connections [1], [2]. However, besides efficient and reliable communication, a wide spectrum of applications also require massive computing capabilities. For example, virtual reality and HD video streaming require a large amount of computing resources for rendering and video encoding/decoding, and the autonomous vehicles rely on computing for artificial intelligence (AI)-based steering control. These computation-intensive applications pose great challenges on the battery and computing capabilities of the resource-constrained end devices, especially the IoT devices, which motivates the cloud computing in which computation-intensive applications are offloaded to the cloud servers with centralized and abundant computation resources. Although cloud computing can significantly reduce the computation delay and the energy consumption of the users, it may fail to meet the demands of delay sensitive applications, such as mobile gaming and augmented reality, since the long transmission distances between end users and the cloud servers result in long transmission delays. To address this issue, mobile edge computing (MEC) has been extensively investigated, where the computing resources in the network edge are employed to provide efficient and flexible computing services. In 5G wireless systems, ultra-dense network edge devices will be deployed, such as macro/small cell base stations and WiFi access points which can provide exponentially growing amount of edge computing resources. Many significant issues in MEC have been extensively investigated, including offloading task model [3], [4], energy efficiency [5]–[7], latency reduction [8]–[10], and joint optimization of communication and computing [11], [12].

However, 5G networks may fail to provide ubiquitous coverage to suburban and rural areas, where IoT devices could be widely deployed to execute certain applications with relatively high computing requirements. For example, the fusion of sensing information, especially the handling of high-definition sound or video information, will quickly drain the battery of the sink nodes and result in large processing delays. Due to the lack of terrestrial access network coverage, the typical edge and cloud computing paradigms cannot be applied in such scenarios. To this end, we propose to employ the space-air-ground integrated network (SAGIN) architecture for the computing offloading of remote IoT applications. SAGIN integrates the satellite network and aerial network

Manuscript received October 10, 2018; revised January 10, 2019 and March 8, 2019; accepted March 13, 2019. Date of publication March 21, 2019; date of current version April 16, 2019. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 91638204 and in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. *(Corresponding author: Wei Quan.)*

N. Cheng is with the School of Telecommunication, Xidian University, Xi'an 710071, China, and also with the Electrical and Computer Engineering Department, University of Waterloo, Waterloo, ON N2L3G1, Canada (e-mail: nancheng@xidian.edu.cn).

F. Lyu, C. Zhou, W. Shi, and X. Shen are with the Electrical and Computer Engineering Department, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: f2lyu@uwaterloo.ca; c89zhou@uwaterloo.ca; w46shi@uwaterloo.ca; sshen@uwaterloo.ca).

W. Quan is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: dr.wei.quan@ieee.org).

H. He is with the School of Information Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: hongli_he@zju.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2019.2906789

with the terrestrial network to provide seamless and flexible network coverage and services to large areas, and thus can be applied in many promising fields, such as intelligent transportation system, remote area monitoring, disaster rescue, and large-scale high-speed mobile Internet access [13]. SAGIN is a multidimensional heterogeneous network consisting three network segments, i.e., the satellite network, aerial network, and terrestrial network. Each network segment possesses different resources and is affected by different limitations. The Low Earth Orbit (LEO) and geostationary (GEO) satellites constitute a hierarchical network where LEO satellites provide high-speed access and GEO satellites relay the data between LEO for long distance transmission [14]. The aerial network, including flying unmanned aerial vehicles (UAVs), high latitude platforms (HAPs), and communication balloons, can be deployed on demand at locations with burst data traffic to offer high-speed and dynamic network services, such as dynamic coverage, edge computing, crowdsensing, etc. [15], [16]. In the proposed SAG-IoT computing offloading architecture, the aerial network nodes can serve as the flying edge servers, which provides the IoT devices with the low-delay edge computing. On the other hand, the satellite communication, although may have lower communication rate and higher transmission delay, can provide always-on cloud computing through seamless coverage and satellite backbone networks [17]. However, employing the SAGIN in IoT computing offloading introduces several challenging issues. Firstly, the high mobility of the aerial network results in dynamic channel conditions and coverage, leading to varying server availability and communication delay, which should be carefully handled to guarantee the performance of the SAG-IoT system. Secondly, different network segments in SAGIN possess distinct network conditions and resource constraints, and it is non-trivial to design an efficient computing offloading approach considering the complex and dynamic network conditions and resources.

In this paper, we present a flexible joint communication and computation SAGIN framework to provide powerful edge/cloud computing services to remote IoT users. Under the framework, we propose an efficient computing offloading approach which learns on-the-fly the optimal offloading policy to minimize the weighted sum of delay, energy consumption, and server usage cost, considering the multidimensional network dynamics and resource constraints. Firstly, the UAV edge servers' computation resources are virtualized as virtual machines (VMs) for parallel execution of the offloaded tasks. We formulate the joint VM resource allocation and task scheduling problem as a mixed-integer programming problem and propose an efficient heuristic algorithm to solve it. Secondly, we investigate the computing offloading problem in SAGIN, which is formulated as a Markov decision process (MDP). To learn the network dynamics, a model-free reinforcement learning (RL)-based approach is proposed, and an actor-critic learning algorithm is designed to handle the large state and action spaces. To the best of our knowledge, our work is the first work to study the computing offloading problem in SAGIN, which validates the feasibility of SAGIN supporting computation-intensive applications for remote IoT users, and can provide useful guidelines for SAGIN network design and remote computing offloading.

The main contributions of the paper can be summarized as follows.

- We formulate the SAG-IoT computing offloading problem as an MDP and propose an RL-based approach to efficiently solve the problem. The system state is defined to integrate the historical network information to learn the system dynamics. In addition, a policy gradient-based actor-critic learning algorithm is proposed to cope with problem of dimensionality curse and accelerate the learning speed.
- We adopt network virtulization to flexibly allocate the resources of the edge server. We formulate the joint edge server VM computation resource allocation and task scheduling problem as a mix-integer programming problem, and propose an effective heuristic algorithm to solve it.
- The performance of the proposed approaches are evaluated through extensive simulations. The joint VM allocation and task scheduling can achieve near-optimal performance with low complexity. In addition, the performance of the proposed RL-based computing offloading approach is evaluated with respect to design parameters.

The remainder of the paper is organized as follows. In Section II, we present the related work. Section III describes the system model. In Section IV, the joint edge VM allocation and task scheduling problem is formulated and solved. Section V formulates the SAG-IoT computing offloading problem, followed by the RL-based solution in Section VI. Section VII evaluates the proposed approaches, and Section VIII concludes the paper. Useful notations used throughout the paper are listed in Table I.

II. Related Work

A. Mobile Edge Computing

The concept of MEC was originally proposed by ETSI in [18], in which the motivation, definition, architecture, and challenging issues are discussed. In edge computing, the computation task offloading mechanism determines the overall performance of the MEC system. The energy-efficient computation offloading is crucial for energy-constraint IoT devices, and has been studied in [5] and [6]. In [5], Mahmoodi *et al.* studied the joint scheduling and computation offloading problem and proposed a real data measurement based optimization method to save the energy consumption of the mobile users. In [6], Mao *et al.* proposed a Lyapunov method-based dynamic computation offloading for devices with energy harvesting. The execution cost which jointly considers the execution latency and task failure is taken as the performance metric. In MEC system, the energy consumption and task delay rely not only on the task processing, but also on the communication of the related data of the task. Therefore, the joint optimization of the communication radio resources and the computing offloading has attracted much research attention [11], [12]. In [11], You *et al.* studied the resource allocation for multiuser MEC offloading problem considering

TABLE I
NOTATIONS USED IN THE PAPER

Notation	Description
M	Number of IoT users
N	Number of IoT applications
$\mathcal{C}^l$, $\mathcal{C}^e$, $\mathcal{C}^c$	The computation resources of local IoT users, the edge servers, and the cloud server
E^l	The local process power consumption of IoT users
E_i^e, E_i^c	The transmission power of IoT users to UAV and satellite
$\mathcal{B}_{ij}^e$, $\mathcal{B}_{ij}^c$	The usage cost of task W_{ij} in edge server and cloud server
B	Bandwidth of UAV-ground communication
H_j^{in}, H_j^{out}	The size of input data and output data of j-th application
Z_j	The computing requirement of j-th application
$M(t)$, $m_{ij}(t)$	Remaining tasks matrix and the element corresponding to task W_{ij}
$X^l(t)$, $X^e(t)$, $X^c(t)$	Offloading decision matrix corresponding to local process, offload to UAV, and offload to the cloud
$x_{ij}^l(t)$, $x_{ij}^e(t)$, $x_{ij}^c(t)$	The element of $X^l(t)$, $X^e(t)$, $X^c(t)$ corresponding to task W_{ij}
α, β	The weight of UAV-edge and cloud server usage cost over the IoT user energy consumption
ϖ_i	The weight of delay over energy consumption and server usage cost for IoT user i

TDMA and OFDMA scenarios. In [12], Wu *et al.* studied the multi-access-assisted computing offloading, and presented a joint optimization of computation task scheduling and radio resource allocation. However, these works only focus on the fixed MEC scenario, i.e., the edge computing services are provided by cellular BSes or WiFi APs, which is different from our work where flying UAVs serve as the mobile edge servers. In [19], a mobile edge computing mechanism is proposed via a UAV-Mounted cloudlet. The bit allocation and UAV trajectory are jointly designed to minimize the mobile energy consumption by solving a non-convex optimization problem. Different from [19], we consider both the energy consumption and task processing delay. In addition, the UAV trajectories are learnt instead of designed for the scenarios where the UAVs are not deployed by network operators and the trajectories are unknown in advance.

B. Space-Air-Ground Integrated Network

SAGIN is envisioned as a promising technology to address many problems in future mobile communication networks, such as remote and large-scale coverage, growth of mobile data, uneven data traffic, and rigid backbone networks, and has recently attracted much attention from both academia and industry. Different SAGIN architecture is discussed in [20], [21]. In [20], Hoang *et al.* studied the optimal energy allocation problem in SAGIN and proposed a learning-based algorithm to optimize the network performance and maximize the service providers' revenue. In [21], Zhang *et al.* proposed a software-defined SAGIN architecture and discussed the challenging issues therein. The edge caching is employed in SAGIN to reduce the content retrieval delay and offload the backbone networks. In [22], Chen *et al.* proposed an optimal content caching scheme to place content at UAVs by considering the user's information and the content request distribution. However, the study of edge computing offloading and computation resource allocation considering the cooperation of space, aerial, and ground network segments is still missing in the literature, which is important for supporting a myriad of computing-intensive applications in SAGIN.

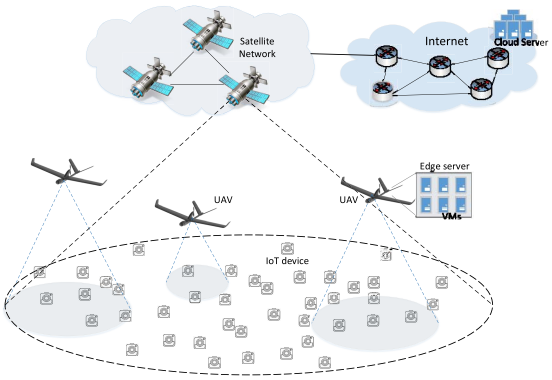


Fig. 1. An overview of the SAG-IoT architecture.

III. SYSTEM MODEL

A. Network Model

We consider a remote area where IoT devices are deployed to conduct certain tasks with computation requirements, such as monitoring and video surveillance. In the considered remote area, there is no cellular coverage, and therefore we consider a space-air-ground integrated network (SAGIN) to provide network functions, such as network access, edge computing and caching, to the IoT devices. The overview of the SAG-IoT network is shown in Fig. 1. In the SAG-IoT network, there are three network segments, i.e., the ground segment, the aerial segment, and the space segment. The IoT devices compose the ground segment and have very limited energy and computing capabilities. The applications running at the IoT devices may generate data to upload and computing tasks to execute. In the aerial segment, the flying UAVs can serve as edge servers to provide ground users with edge caching and computing capabilities. The flying UAVs, such as the Facebook Aquila, can fly for months without charging by using solar panels [23]. The UAVs are configured with fixed flying trajectories to serve the considered area. Furthermore, in the space segment, one or more LEO satellites provide the full coverage of the area of interest, and connect the IoT devices with the cloud servers through the satellite backbone network.

For the IoT device (user) i, it has the local computing capability of $\mathcal{C}^l$, which is assumed identical for all users. The energy consumption for locally task computing/processing is denoted by E^l, which is related to $\mathcal{C}^l$. The power consumption for transmission to UAV and satellite is denoted by E_i^e and E_i^c, respectively. In the edge servers, i.e., UAVs, the computing resources are virtulized as VMs, each for one specific application [24]. In edge server k, the total computation resource is $\mathcal{C}^e$, the resources allocated to the computation VM v is denoted by $\mathcal{C}_v^e$, and the server usage cost of the computation VM for user i's task j is denoted by $\mathcal{B}_{i,j}^e$. For the UAV-ground communication, since we consider the task offloading decision making, which is with much longer time scale than traditional resource scheduling time (1 ms), only large-scale channel fading is considered. In addition, since the instantaneous channel information is not required, a satellite controlled global decision making is feasible. According to [25], the pathloss between the UAV and the ground users follows

$$L(r,h) = 20\log\big(\frac{4\pi f_c(h^2+r^2)^{\frac{1}{2}}}{c}\big) + P_{LoS}(r,h)\eta_{LoS} + (1-P_{LoS}(r,h))\eta_{NLoS}, \quad (1)$$

where h and r denote the UAV flying altitude and the horizontal distance between the UAV and the ground user, respectively. η_{LoS} and η_{NLoS} denote respectively the additive loss incurred on top of the free space pathloss for LoS and NLoS links [26]. f_c denotes the carrier frequency, and c denotes the speed of light. P_{LoS} is the line-of-sight probability of UAV-ground link, which can be calculated by

$$P_{LoS}(r,h) = \frac{1}{1+a\exp(-b(arctan(\frac{h}{r})-a))}. \quad (2)$$

$(a, b, \eta_{LoS}, \eta_{NLoS})$ are environment-dependent variables. For instance, in remote areas, their values are (4.88, 0.43, 0.1, 21) [27]. In addition, the UAV-ground communication uses WiFi protocols with total bandwidth B. If n IoT devices communicate with a UAV simultaneously, the bandwidth each IoT device obtains is calculated by

$$B_i = \rho B\xi(n) \quad (3)$$

where ρ is the WiFi throughput efficiency factor, and $\xi(n)$ is the WiFi channel utilization function which is a decreasing function of contenting user number n. Thus, the instant UAV-ground and ground-UAV data rate can be calculated by

$$r_{GU} = \rho B\xi(n)\log_2(1+\frac{E_i^e 10^{-L_i/10}}{\sigma^2}), \quad (4)$$

and

$$r_{UG} = \rho B\xi(n)\log_2(1+\frac{E_i^{e-} 10^{-L_i/10}}{\sigma^2}), \quad (5)$$

respectively, where E_i^{e-} denotes the UAV transmit power to ground IoT users, L_i denotes the pathloss for the corresponding IoT user-UAV link, and σ^2 denotes the power of the Gaussian noise. For the satellite-ground communication, we consider a constant communication data rate r_{SG}, which is usually smaller than the UAV-ground date rate. The satellite is connected to the Internet/cloud through the satellite backbone network. We denote the transmission rate between the satellite and the cloud by r_{SC}. The cloud has much higher computing capability than IoT devices and edge servers, and the processing rate for each task is denoted by $\mathcal{C}^c$, and the usage cost for user i's task j is denoted by $\mathcal{B}_{i,j}^c$.

B. Multi-User Multi-Task SAG-IoT Computing Offloading

We consider that there are M IoT users and N different computation applications, and each user is running all N applications, leading to $M \times N$ computation tasks in the system. We also consider that the N applications have certain priorities, in the way that if multiple tasks are scheduled simultaneously, the task with smaller application number will be transmitted/processed earlier than those with larger application numbers. For j-th application, the size of the input data, the output data, and the workload are denoted by H_j^{in}, H_j^{out} and Z_j, respectively. These tasks can be executed locally at the IoT devices. However, due to the limited energy and computing capability of IoT devices, the computing tasks can also be offloaded to the UAV edge servers or further to the cloud through the satellites. The offloading decision is made in each time slot until all the $M \times N$ tasks are completed. At the beginning of time slot t, the remaining tasks are denoted by a $M \times N$ matrix $\mathbf{M}(t)$, where the element $m_{i,j}(t) = 1$ indicates task W_{ij} has not completed, and $m_{i,j}(t) = 0$ otherwise. Denote decisions of locally processing the tasks, offloading the tasks to edge, and offloading the tasks to cloud at time slot t by $M \times N$ matrices $\mathbf{X}_l(t)$, $\mathbf{X}_e(t)$, and $\mathbf{X}_c(t)$, respectively, and each binary element $x_{ij}^l(t)$, $x_{ij}^e(t)$, and $x_{ij}^c(t)$ indicates whether task W_{ij} is processed locally, offloaded to the edge, or offloaded to the cloud, respectively. Note that task W_{ij} can be scheduled to at most one means at time t, i.e., the offloading decision is constrained by

$$x_{ij}^l(t), x_{ij}^e(t), x_{ij}^c(t) \in \{0,1\}, \quad (6)$$

$$x_{ij}^l(t) + x_{ij}^e(t) + x_{ij}^c(t) \le m_{ij}(t). \quad (7)$$

The inequality in (7) holds when an unfinished task is not scheduled at time slot t. If the task W_{ij} is processed locally or offloaded to the cloud at time t, we consider the task can be finished with a certain delay, and $m_{i,j}(t+1) = 0$. However, if W_{ij} is offloaded to the UAV edge server, it may not be completed and return to user i successfully at the end of t, which is due to two reasons. Firstly, if multiple tasks are offloaded to one UAV edge server, some of them may not be able to be completed within the time slot; secondly, since the UAVs are moving, when task W_{ij} is completed in the edge server, the result cannot be transmitted to user i if user i is out of the coverage area of the UAV.

C. Cost Model

The computing task offloading is to minimize the system cost of executing the $M \times N$ tasks. In the considered SAG-IoT system, the system cost is composed of two parts, i.e., the delay cost and the energy and server usage cost.

1) Delay Cost: If the task W_{ij} is scheduled at time slot t, the delay can be calculated according to the offloading decision. If the task is scheduled to process locally, the delay is

$$T_{ij}^l = \varepsilon(t-1) + t_{r,i}^l + \frac{Z_j}{\mathcal{C}^l}, \quad (8)$$

where ε is the length of the time slot, and $\varepsilon(t-1)$ is the elapsed time since the generation of the task. Due to the low computing capability of IoT devices, it is likely that at the beginning of time slot t, there are some tasks which are scheduled to locally process yet not finished. $t^l_{r,i}$ is the time for user i to complete the remaining local processing tasks, which can be calculated by the remaining local workload divided by the local processing capability $\mathcal{C}^l$. If the task is offloaded to the UAV edge server, and the result is returned to user i within time slot t, the total delay of the task can be calculated by

$$T^e_{ij} = \varepsilon(t-1) + d^e_{ij} + \frac{\sum_{a=1}^{j} x^e_{i,a}(t) H^{in}_a}{r_{GU}} + \frac{H^{out}_j}{r_{UG}}. \quad (9)$$

where d^e_{ij} denotes the processing delay of W_{ij} in the UAV edge server, which depends on the offloading decision and VM resource allocation in the server as described in Section IV. If multiple tasks of user i are scheduled to the edge server, $\sum_{a=1}^{j} x^e_{ia}(t) H^{in}_a$ calculates the time for transmitting W_{ij} task data to the server considering the transmission of tasks with higher priorities. Similarly, if the task is offloaded to the cloud through the satellite, the delay is calculated by

$$T^c_{ij} = \varepsilon t + \frac{Z_j}{\mathcal{C}^c} + \frac{H^{in}_j + H^{out}_j}{r_{SG}} + \frac{H^{in}_j + H^{out}_j}{r_{SC}}. \quad (10)$$

2) Energy and Server Usage Cost: The energy cost of locally processing W_{ij} can be calculated by

$$L^l_{ij} = E^l \frac{Z_j}{\mathcal{C}^l} \quad (11)$$

If at time slot t, task W_{ij} is offloaded to the UAV edge server and the result is successfully transmitted to user i, the energy and server usage cost can be calculated by

$$L^e_{ij} = E^e_i \sum_{t=1}^{v} x^e_{ij}(t) \frac{H^{in}_j}{r_{GU}(t)} + \alpha \mathcal{B}^e_{ij}, \quad (12)$$

where α represents the weight of the UAV server usage cost over the IoT user energy consumption. $\sum_{t=1}^{v} x^e_{ij}(t) \frac{H^{in}_j}{r_{GU}(t)}$ calculates the total energy consumption considering the case in which former times of offloading of the task to a UAV edge server failed to return within the scheduling slot. Similarly, if task W_{ij} is offloaded to the cloud, the energy and server usage cost can be calculated by

$$L^c_{ij} = \frac{E^c_i H^{in}_j}{r_{SG}} + \beta \mathcal{B}^c_{ij}, \quad (13)$$

where β denotes the weight of cloud server usage cost over the IoT user energy consumption.

IV. Computation VM Allocation

In time slot t, multiple tasks may be offloaded to one UAV edge server. In such a scenario, these tasks are executed in different VMs in parallel to reduce the processing latency. One VM executes the tasks belonging to a specific application. We therefore study a VM allocation problem to allocate the edge server computation resources to different VMs considering the tasks offloaded to the edge server. In addition, due to the mobility of UAVs, some users may lose connection with the UAV quickly, and thus executing such tasks may lead to excessive resource allocated to the corresponding VM. For example, in Fig. 2, two VMs are considered to execute the offloaded tasks to the UAV edge server, and $t_{i,j}$ is the delay requirement of task j in VM i. We can see that the delay requirement $t_{2,1}$ is very strict and a larger amount of computation resource should be allocated to VM2 to finish the corresponding task before deadline. However, since the total computation resource of an edge server is fixed, it is likely that little resource allocated to VM1, and none of the three tasks in VM1 can be finished in time. Therefore, we jointly optimize the VM allocation and task scheduling in the UAV edge server to reduce the system sum delay.

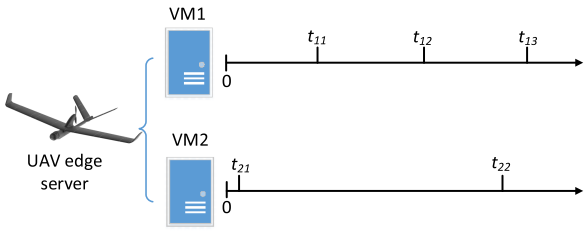


Fig. 2. An example of joint VM allocation and task scheduling for UAV edge server.

In the considered problem, there are multiple kinds of applications (Apps), denoted by $\mathcal{A} = \{1, \ldots, N\}$, and one UAV edge server with computation capability $\mathcal{C}$ cycles/s.[1] For m-th App, there might be multiple offloaded tasks, denoted by $\mathcal{T}_m = \{1, \ldots, N_m\}$, which has same computation workload but different maximum delay requirements. Note that Z_m denotes the computation workload of m-th App's tasks. $\mathbf{C} = \{c_m \mid m \in \mathcal{A}\}$ denotes the computation resource variables, where c_m is the computation resource allocated to the VM executing App m. $\mathbf{Y} = \{y_{m,n} \mid m \in \mathcal{A}, n \in \mathcal{T}_m\}$ denotes the decision variables on task execution, where $y_{m,n} = 1$ if task n of App m is scheduled and executed, and $y_{m,n} = 0$ otherwise. Therefore, our sum delay minimization problem can be formulated as follows.

$$\begin{aligned}
\min_{\mathbf{C},\mathbf{Y}} \ & \sum_{m=1}^{N} \sum_{n=1}^{N_m} \left[y_{m,n} \sum_{k=1}^{n} y_{m,k} \frac{Z_m}{c_m} + \varepsilon \left(1 - y_{m,n}\right) \right] \\
\text{s.t. C1:} \ & \sum_{k=1}^{n} y_{m,k} \frac{Z_m}{c_m} \leqslant t_{m,n}, \quad \forall m \in \mathcal{A}, \ \forall n \in \mathcal{T}_m \\
\text{C2:} \ & \sum_{m=1}^{M} c_m \leqslant \mathcal{C} \\
\text{C3:} \ & c_m \geqslant 0 \\
\text{C4:} \ & y_{m,n} \in \{0, 1\}, \quad \forall m \in \mathcal{A}, \ \forall n \in \mathcal{T}_m
\end{aligned} \quad (14)$$

where $t_{m,n}$ is the delay requirement of task n of App m and ε is the length of the time slot. $t_{m,n}$ can be calculated by

$$t_{m,n} = \min(t_{lc}, \varepsilon) \quad (15)$$

where t_{lc} is the time when the user who offloads this task loses connection with the UAV. C1 restricts the maximum delay for

[1]Here we use $\mathcal{C}$ instead of $\mathcal{C}^e$ for simplicity.

each task if it is executed at current time slot. C2 limits that the total computation resources of VMs cannot exceed $\mathcal{C}$.

It can be seen that Problem (14) is a mixed-integer programming that is difficult to solve. It involves the continuous variable $\mathbf{C}$ and 0-1 integer variable $\mathbf{Y}$. Even though we assume $\mathbf{C}$ is known, the residual subproblem is still a quadratic problem with 0-1 integer constraints, which is NP-hard with non-definite matrix [28], [29]. This problem is commonly reformulated by specific relaxation approach and then solved by powerful convex optimization techniques. However, this method performs extensive iterations and reveals little insight about scheduling policy. Thus, we are motivated to design an efficient low-complexity algorithm to obtain the suboptimal solution. In the proposed VM allocation and task scheduling algorithm, we assume for each VM m, the delay requirements for N_m tasks have been sorted, i.e., $t_{m,n} \leq t_{m,n+1}$. At the beginning, we try to allocate c_m as if all tasks had been scheduled, i.e., $y_{m,n} = 1$, $\forall m \in \mathcal{A}, \forall n \in \mathcal{T}_m$. The allocation results would be

$$c_m = \min\{\frac{nZ_m}{t_{m,n}}\}, \quad \forall m \in \mathcal{A},\ \forall n \in \mathcal{T}_m. \tag{16}$$

Given the allocation results, if $\sum_{m=1}^{M} c_m > \mathcal{C}$, it means not all tasks can be scheduled. Therefore, we choose not to schedule the task with the most harsh delay requirement, i.e., let

$$y_{m,n} = 0, \tag{17}$$

where

$$m, n = \arg\max_{m,n} \ \frac{nZ_m}{t_{m,n}}, \quad \forall m \in \mathcal{A},\ \forall n \in \mathcal{T}_m. \tag{18}$$

Then, we calculate the VM allocation c_m again. Repeat this process until the condition $\sum_{m=1}^{M} c_m \leq \mathcal{C}$ is satisfied, and the VM allocation c_m and task scheduling $\mathbf{Y}$ is obtained. Note that for a generic $\mathbf{Y}$, the VM allocation is calculated by

$$c_m = \min\{\frac{\sum_n y_{m,n} Z_m}{t_{m,n}}\}, \quad \forall m \in \mathcal{A},\ \forall n \in \mathcal{T}_m, \tag{19}$$

and the unscheduled task selection is calculated by

$$m, n = \arg\max_{m,n} \ \frac{\sum_n y_{m,n} Z_m}{t_{m,n}}, \quad \forall m \in \mathcal{A},\ \forall n \in \mathcal{T}_m. \tag{20}$$

The full algorithm of edge server VM allocation and task scheduling is shown in Algorithm 1. From the algorithm, we can see that the worst case (the cloud cannot finish any offloaded task in time) requires $N'(N'+1)/2$ comparisons where N' is the number of total offloaded tasks to the UAV edge server. Even the worst case complexity $O(N'^2)$ is very low, and therefore the proposed algorithm can work efficiently in the dynamic SAGIN environment.

Algorithm 1 VM Allocation and Task Scheduling in Edge Server

1: **Input**: $\mathcal{C}$, $\mathcal{T}_m$, $t_{m,n}$, ε.
2: **Output**: VM allocation c_m, task scheduling $\mathbf{Y}$.
3: Initialize $y_{m,n} = 1$ $\forall m, n$, and c_m according to (19).
4: **while** $\sum_{m=1}^{M} c_m > \mathcal{C}$ **do**
5: Update $y_{m,n}$ according to (17) and (20).
6: Update c_m according to (19).
7: **end while**
8: **return**

V. Computation Offloading Problem Formulation

We design an online computing offloading approach for the SAG-IoT system, in which at each time slot the computing tasks of IoT devices are scheduled to process locally, offloaded to the UAV edge server, or offloaded to the cloud server through the satellite, in order to minimize the total system cost in terms of the delay of the tasks, the energy consumption of the IoT users, and the edge and cloud server usage costs. This can be achieved by modeling the computing offloading decisions as an MDP.

An MDP is defined by a tuple (S, A, T, R), where S is the set of possible system states, A is the set of actions, $T = \{p(s'|s,a)\}$ is the set of transition probabilities, and $R : S \times A \mapsto \Re$ is a real-value reward (or cost) function when the system is at state $s \in \mathbf{S}$ and an action $a \in A$ is taken. A policy π is a mapping from S to A. The MDP of the SAG-IoT computing offloading problem is defined as follows.

1) States: at the beginning of time slot t, the network state is defined as $\mathbf{M}(t) \otimes \mathbf{T}^r(t) \otimes \mathbf{PL}(t) \otimes \mathbf{PL}(t-1) \otimes \mathbf{PL}(t-2) \otimes \cdots \otimes \mathbf{PL}(t-t_q)$, where $\mathbf{T}_r^l(t) = \{t_1^l(t), t_2^l(t), \ldots t_M^l(t)\}$ represents the remaining time for each user to complete locally processing tasks, and $\mathbf{PL}(t) = \{PL_1(t), PL_2(t), \ldots, PL_M(t)\}$ is the vector of pathloss values of all users to their associated UAV. The system state includes the pathloss information of the current and the previous t_q time slots in order to learn and predict the pathloss information.

2) Actions: at the beginning of time slot t, the system takes the action of scheduling the tasks of the users, i.e., to determine the matrices $\mathbf{X}_l(t)$, $\mathbf{X}_e(t)$, and $\mathbf{X}_c(t)$, or equally, to determine x_{ij}^l, x_{ij}^e, and x_{ij}^c, $\forall i, j$. Therefore, we denote $a(t) = \{\mathbf{X}_l(t), \mathbf{X}_e(t), \mathbf{X}_c(t)\}$. Clearly, at time slot 0, there are 4^{MN} possible actions, which is a very large number when M and N are large.

3) Transition probability: since the UAV-user pathloss is not affected by the actions, the system transition probability can be calculated by

$$p(s_{t+1}|s_t, a_t) = p(\mathbf{PL}(t+1)|\mathbf{PL}(t)) \cdot (\mathbf{T}_r^l(t+1)|\mathbf{T}_r^l(t), a_t) \cdot p(\mathbf{M}(t+1)|\mathbf{M}(t), a_t). \tag{21}$$

Specifically, if the UAV trajectory and the flying speed are planned to be fixed, $p(\mathbf{PL}(t+1)|\mathbf{PL}(t))$ is 1 with a specific $\mathbf{PL}(t+1)$ and 0 otherwise. However, due to the uncertainties in the UAV mobility, $p(\mathbf{PL}(t+1)|\mathbf{PL}(t))$ will be difficult to model. $\mathbf{T}_r^l(t+1)$ can be calculated by

$$T_{r,i}^l(t+1) = \max\{T_{r,i}^l(t) + \sum_{j=1}^{N} x_{ij}^l(t) \frac{Z_j}{\mathcal{C}^l} - \varepsilon, 0\}. \tag{22}$$

For $p(\mathbf{M}(t+1)|\mathbf{M}(t), a_t)$, it is difficult to model accurately. For example, if a task is offloaded to a UAV edge server, whether the task can be complete within the time slots depends on the UAV data transmission rate, UAV computation resource

allocation, other users' decision, and UAV mobility, which are dynamic and correlated.

4) Reward: To minimize the weighted sum of delay, energy, and server usage cost, we use the cost function $C(s_t, a_t) = \sum_{ij} C_{i,j}(s_t, a_t)$ at time slot t, where $C_{i,j}(s_t, a_t)$ is the cost function of task W_{ij}, which is calculated in the following way.

1) if $m_{ij}(t) = 0$, the task has already completed, and thus $C_{ij}(s_t, a_t) = 0$.
2) if $m_{ij}(t) = 1$ and $x^l_{ij} + x^e_{ij} + x^c_{ij} = 0$, the task is not scheduled in this time slot, and thus a delay of ε is introduced. We define the cost function $C_{ij}(s_t, a_t) = \varpi_i \varepsilon$, where ϖ_i is user i's weight on the delay.
3) if $m_{ij}(t) = 1$ and $x^l_{ij} + x^e_{ij} + x^c_{ij} = 1$, $C_{ij}(s_t, a_t) = \varpi_i(x^l_{ij}(T^l_{ij} - \varepsilon t) + x^e_{ij}(T^e_{ij} - \varepsilon t) + x^c_{ij}(T^l_{ij} - \varepsilon t)) + x^l_{ij} L^l_{ij} + x^e_{ij} L^e_{ij} + x^c_{ij} L^l_{ij}$.

Define the value function V of state s as the expected long-term discounted cost starting from s with policy π, i.e.,

$$V(s|\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t) | s_0 = s, \pi\right], \tag{23}$$

where $\gamma \in [0, 1]$ is a discount factor, and the expectation is taken over all possible state trajectories starting from s. The online computing offloading approach is to select an optimal policy π^*, which minimizes the value function of each state, i.e.,

$$\pi^*(s) = \arg\min_a \sum_{s'} p(s'|s, a)[C(s, a) + \gamma V(s'|\pi^*))]. \tag{24}$$

VI. RL-Based Offloading Decision Making

In problem (24), the reward function and transition probabilities are difficult to model accurately due to the UAV mobility and dynamic VM allocation of UAV edge servers. In addition, with the increasing system scale, i.e., M and N, the exponentially growing system state space makes the system intractable. Therefore, the proposed online computing offloading problem can be solved by model-free RL-based methods, such as Q-learning [30] and policy gradient methods [31]. Although Q-learning methods have shown great potentials in solving RL problems with a large state space, it usually cannot efficiently deal with problems with large or even continuous action spaces, which is the case in problem (24). Therefore, in this paper, we propose an online computing offloading approach for the SAG-IoT system by adapting the policy gradient method.

In the proposed online computing offloading approach, the policy is parameterized by a vector $\boldsymbol{\theta} \in \mathfrak{R}^d$, i.e., $\pi(a|s, \boldsymbol{\theta}) = P(a_t = a|s_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta})$, for the probability that action a is taken when the system is in state s at time t, under the policy with parameter $\boldsymbol{\theta}$. If $\boldsymbol{\theta}$ is defined for each feature of the state, i.e., each element in $M(t)$, $T^r(t)$, and $PL(t)$, the length of vector $\boldsymbol{\theta}$ is $M(N + t_q + 2)$. To learn the policy parameter, we first define the performance measure of $\boldsymbol{\theta}$, which is denoted by $J(\boldsymbol{\theta})$. Since the online computing offloading problem is episodic (an episode ends when all MN tasks are finished), we define the performance measure as the total discounted cost of the episode of computing all tasks. Denote by τ a trace of state-action sequence $s_0, a_0, s_1, a_1, s_{t_{max}}, a_{t_{max}}$ in an episode following $\pi(\cdot|\cdot, \boldsymbol{\theta})$, where t_{max} denotes the preset value indicating the possible maximum number of time slots for processing all tasks. Then, we can have $J(\boldsymbol{\theta})$ as the value function of the start state s_0:

$$J(\boldsymbol{\theta}) \doteq V_{\pi_{\boldsymbol{\theta}}}(s_0) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\sum_{k=0}^{t_{max}} \gamma^k C(s_k, a_k) | \pi(\cdot|\cdot, \boldsymbol{\theta})]. \tag{25}$$

To learn the policy parameter $\boldsymbol{\theta}$ which minimizes $J(\boldsymbol{\theta})$, intuitively, we can use the gradient descent method to gradually update $\boldsymbol{\theta}$ by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \varphi \nabla J(\boldsymbol{\theta}_t). \tag{26}$$

where φ represents the learning rate. According to the policy gradient theorem [32], we have

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}_t) &= \mathbb{E}_\pi[\sum_a q_\pi(s_t, a) \nabla_{\boldsymbol{\theta}} \pi(a|s_t, \boldsymbol{\theta})] \\
&= \mathbb{E}_\pi[\sum_a \pi(a|s_t, \boldsymbol{\theta}) q_\pi(s_t, a) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|s_t, \boldsymbol{\theta})}{\pi(a|s_t, \boldsymbol{\theta})}] \\
&= \mathbb{E}_\pi[q_\pi(s_t, a_t) \frac{\nabla_{\boldsymbol{\theta}} \pi(a_t|s_t, \boldsymbol{\theta})}{\pi(a_t|s_t, \boldsymbol{\theta})}] \\
&= \mathbb{E}_\pi[G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(a_t|s_t, \boldsymbol{\theta})}{\pi(a_t|s_t, \boldsymbol{\theta})}].
\end{aligned} \tag{27}$$

Note that $q_\pi(s, a)$ is the state-action value function for policy π, and $G_t = C_t + \gamma C_{t+1} + \gamma^2 C_{t+2} \ldots$ is the discounted return of cost. Using the above, we can then update $\boldsymbol{\theta}$ by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \varphi G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(a_t|s_t, \boldsymbol{\theta})}{\pi(a_t|s_t, \boldsymbol{\theta})}. \tag{28}$$

However, although such a update method (which is referred to as REINFORCE method [33]) can converge to a local minimum asymptotically, it usually leads to high variance and learns slowly. In the online SAG-IoT computing offloading, both state space and action space are large, and therefore REINFORCE method may not be suitable. To further improve the learning performance, we thus employ the *actor-critic* method in which the approximations to both policy and value functions are learned [34]. In actor-critic method, the policy is updated in each time slot instead of every episode of the computing offloading. Therefore, the number of samples required to learn the optimal policy can be reduced dramatically, which accelerates the learning process. To achieve this, we need to learn the value function and use it as a critic to guide the update of policy at each time slot. Specifically, denote by $\hat{V}(s_t, \boldsymbol{\omega})$ the estimation of the value function of state s_t, where $\boldsymbol{\omega} \in \mathfrak{R}^m$ is the parameter vector to fit the value function. Then, in each time slot t, the update of $\boldsymbol{\theta}$ can be done by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \varphi(C_t + \gamma \hat{V}(s_{t+1}, \boldsymbol{\omega}) - \hat{V}(s_t, \boldsymbol{\omega})) \frac{\nabla_{\boldsymbol{\theta}} \pi(a_t|s_t, \boldsymbol{\theta})}{\pi(a_t|s_t, \boldsymbol{\theta})}. \tag{29}$$

Note that in each time slot, the parameter vector $\boldsymbol{\omega}$ of the estimated value function $\hat{V}$ is also updated according to

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \varphi' \nabla_{\boldsymbol{\omega}} L(\boldsymbol{\omega}), \tag{30}$$

where φ' is the learning rate, and the loss function $L(\boldsymbol{\omega})$ is defined as

$$L(\boldsymbol{\omega}) = |\hat{V}(s_t, \boldsymbol{\omega}) - (C_t + \gamma \hat{V}(s_{t+1}, \boldsymbol{\omega}))|^2. \quad (31)$$

Finally, motivated by the capability of deep neural networks to approximate complex functions, we employ deep learning architecture to learn the policy in terms of $\boldsymbol{\theta}$ and the estimated state-value function. The full proposed online computing offloading approach for SAT-IoT is shown in Algorithm 2, where φ and φ' are learning rates for the actor and the critic, respectively.

Algorithm 2 Deep Actor-Critic Based Online Computing Offloading

Input: IoT user information: location, $\mathcal{C}^l$, E^l, E_i^e, E_i^{e-} and E_i^c
UAV edge server information: mobility traces, $\mathcal{C}_v^e$, $\mathcal{B}_{ij}^e$, B
Cloud related information: r_{SG}, r_{SC}, $\mathcal{C}_c$, $\mathcal{B}_{ij}^c$
Task information: H^{in}, H^{out}, Z
Output: Optimal computing offloading decision $X(t)$
Randomly initialize critic network $\hat{V}(s, \boldsymbol{\omega})$ and actor network $\pi(s, a|\boldsymbol{\theta})$
for episode $= 1$, G **do**
  Initialize a random vector $\mathcal{N}$ as the noise for action exploration
  Observe the initial state s_1
  for time slot t $= 1$, t_{max} **do**
    select action $a_{r,t} = \pi(s|\boldsymbol{\theta}) + \mathcal{N}_t$
    execute a_t and observe the cost C_t and state s_{t+1}
    $\eta \leftarrow C_t + \gamma \hat{V}(s_{t+1}, \boldsymbol{\omega}) - \hat{V}(s_t, \boldsymbol{\omega})$
    update $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \varphi' \eta \nabla_{\boldsymbol{\omega}} \hat{V}(s_t, \boldsymbol{\omega})$
    update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \varphi \eta \gamma^t \frac{\nabla_{\boldsymbol{\theta}} \pi(a_t|s_t, \boldsymbol{\theta})}{\pi(a_t|s_t, \boldsymbol{\theta})}$
  end for
end for
return

The implementation of the proposed RL-based offloading approach is shown in Fig. 3, which is composed of the SAGIN environment, the computing offloading reward evaluator, an actor network, a critic network, and a temporal-difference component. The system state can be observed from the current SAGIN environment, which is then sent to the input of the actor network and the critic network. The actor network generates the action a according to $a = \pi_{\boldsymbol{\theta}}(s)$, and updates the policy $\boldsymbol{\theta}$. It can be easily seen that at time slot t, for an arbitrary task W_{ij}, the decision $x_{ij}(t)$ has four possibilities, i.e., not scheduled, process locally, offload to edge, and offload to cloud. Therefore, we map these four possible decisions of to x_{ij} integer 0, 1, 2, 3 respectively, and design two output layers of the actor network, i.e., $\boldsymbol{\sigma}$ and $\boldsymbol{\mu}$, which can compose $M \times N$ normal distributed random variables to represent the actions of each task. The critic network estimates the value function $\hat{V}(s_t, \boldsymbol{\omega})$ and updates the parameter $\boldsymbol{\omega}$. The reward of a state-action pair is evaluated by the reward evaluator, and is used to calculate the temporal-difference (TD) $\eta = C_t + \gamma \hat{V}(s_{t+1}, \boldsymbol{\omega}) - \hat{V}(s_t, \boldsymbol{\omega})$, which is used in the update of the policy parameter $\boldsymbol{\theta}$ and the critic network parameter $\boldsymbol{\omega}$.

TABLE II
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
M	30	N	5
$\mathcal{C}^l$	200 MC/s	E^l	141 mW
$\mathcal{C}^e$	3 GC/s	E^e, E^{e-}, E^c	200 mW
$\mathcal{C}^c$	10 GC/s	B	20 MHz
h	90 m	r_{SG}	10 Mbps
α	10^{-10} J/cycle	r_{SC}	10 Mbps
β	4×10^{-10} J/cycle	ϖ_i	0.2 J/s

VII. PERFORMANCE EVALUATION

A. Simulation Configurations

In this section, we evaluate the proposed joint VM resource allocation and task scheduling scheme for the UAV edge server, and the RL-based online computing offloading approach for SAG-IoT system. In the simulation, we consider a remote 1 km $\times$ 1 km square area with $M = 30$ IoT users fixed deployed in this area. The IoT user runs $N = 5$ different applications and thus each user has 5 tasks to process. We select ARM Cortex-M based IoT devices as the ground users. Referring to [35] and [36], we set the IoT device computing capability $\mathcal{C}^l$ to 200 MC/s (MC $= 10^6$ cycles), and the energy consumption for local task processing is 141 mW. As defined in [37], the transmission and reception power of IoT users with UAVs and satellites, i.e., E^e, E^{e-}, and E^c are set to 200 mW. 5 UAVs are serving as the flying edge servers for the IoT computing. UAV movement trajectories are planned to maximize the minimum throughput which follows Wu *et al.*'s work [38] with adopting practical UAV-ground propagation channels (1). Referring to [39], the edge server's computation resource $\mathcal{C}^e$ is set to 3 GC/s (GC $= 10^9$ cycles), while the cloud server's computation resource assigned to each task, i.e., $\mathcal{C}^c$, is set to 10 GC/s. For satellite and remote cloud, we consider within one episode of computing offloading, there is one LEO satellite providing the full coverage to the area, and the satellite-ground communication rate r_{SG} is set to 10 Mbps which is the average observed transmission rate in the high throughput satellite communication system ViaSat-1. The satellite-cloud data rate is also constrained by the satellite-ground transmission rate, and therefore we set $r_{SC} = r_{SG} = 10$ Mbps. Different computation tasks may have different computation to data ratios; however, for the simulation simplicity we choose x264 VBR encode computation to data ratio, which is 1300 cycles/byte, i.e., $Z = 1300 H^{in}$ [40]. H_j^{in} and H_j^{out} are randomly chosen between 5 MB and 15 MB, and between 1 MB and 5 MB, respectively. We set the usage cost of edge server/cloud server, i.e., $\mathcal{B}_{ij}^e$ and $\mathcal{B}_{ij}^c$ to the CPU cycles to execute tasks W_{ij}, i.e., W_{ij}'s workload. In addition, $\alpha = 10^{-10}$ J/cycle, $\beta = 4 \times 10^{-10}$ J/cycle, and $\varpi_i = 1$ J/s for each user i. The detailed simulation parameters are shown in Table. II unless otherwise specified.

B. VM Computing Resource Allocation and Task Scheduling

We first evaluate the proposed VM computing resource allocation and task scheduling algorithm. We compare the

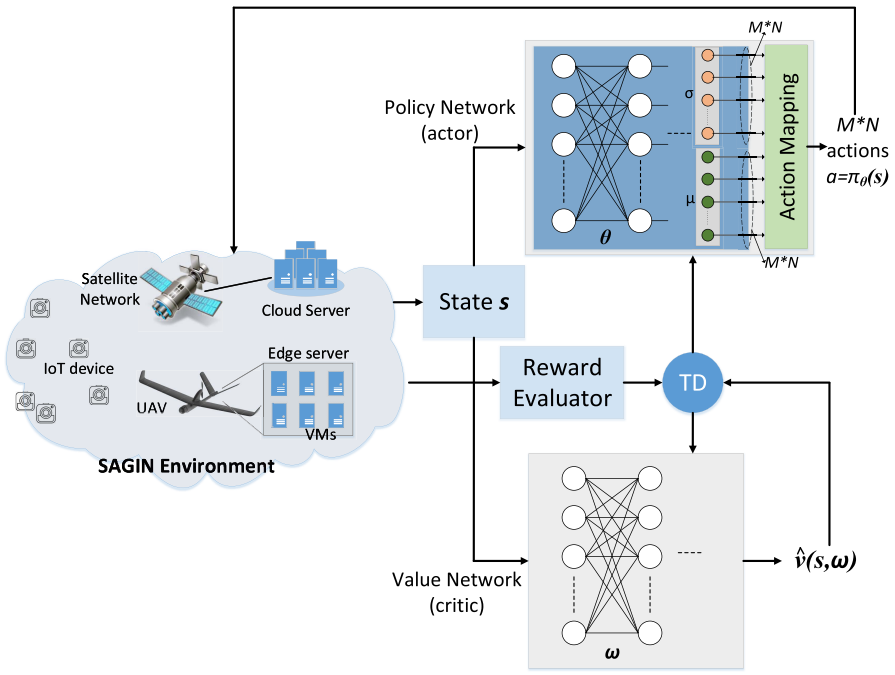


Fig. 3. RL-based computing offloading approach. The proposed approach implements two components, i.e., one actor network to update the policy, and one critic network to evaluate the value function and guide the update of offloading policy.

heuristic algorithm with 'Brute-force' method and 'Random' method. In 'Brute-force', exhaustive search is used to find the optimal unscheduled tasks, which achieves the upper-bound performance but is with high computing complexity. In 'Random', unscheduled tasks are randomly selected.

Fig. 4 shows the delay performance of the proposed algorithm. In Fig. 4(a), the average delay with respect to the UAV edge server computing resource $\mathcal{C}^e$ is shown. We can see from the figure that with the increase of $\mathcal{C}^e$, the average delay of the three methods decrease, because with higher computing server capability, the average processing time will be reduced, and thus more tasks can be scheduled to satisfy their delay requirements. In Fig. 4(b), the average delay with respect to the total number of tasks offloaded to the considered edge server is shown, when $\mathcal{C}^e$ is set to 10 GC. It can be seen that a larger number of tasks lead to increasing average delay, since more tasks contend for the limited computing resources, and fewer tasks can complete in time. In both figures, the proposed heuristic algorithm can achieve a very close performance with that of the 'Brute-force' method, which demonstrates the efficiency of the proposed algorithm.

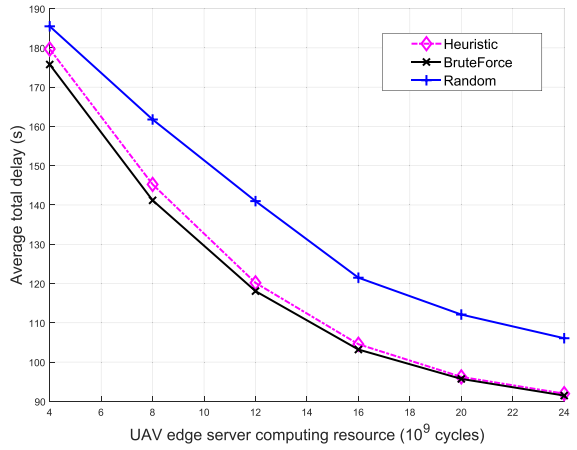
Fig. 5 shows the comparison of the run time between the proposed heuristic algorithm and the 'Brute-force' method. We can see from the figure that with the increasing number of total tasks, the run time of 'Brute-force' method increases exponentially. This is because 'Brute-force' method uses exhaustive search and a larger number of tasks leads to an exponentially growing searching space. In opposite, the run time of the proposed heuristic algorithm remains very small when the number of tasks increases. The zoomed-in run time for the proposed heuristic algorithm shows clearly a quadratic increase on run time when the number of offloaded tasks increase, which validates our analysis in Section IV. To summarize, the proposed VM computing resource allocation and task scheduling algorithm can simultaneously achieve near-optimal performance and very low computational complexity, and therefore is suitable to allocation UAV edge server resources under dynamic network conditions.

C. Deep RL-Based IoT Computing Offloading

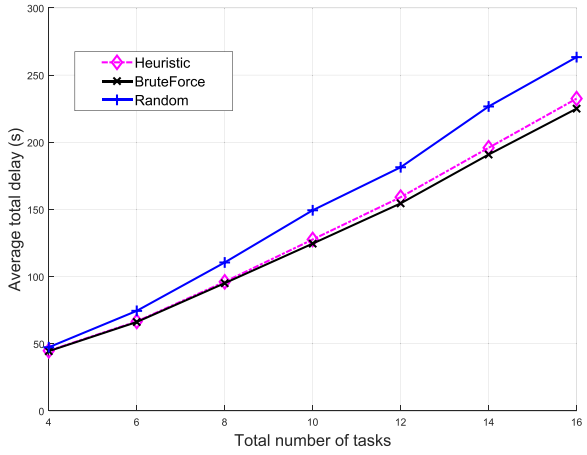
In this part, the performance of the proposed RL-based SAG-IoT computing offloading approach is evaluated and compared. To show the efficiency of our proposed approach, we explicitly compare it with two other computing offloading approaches, i.e., 'Random' and 'Greedy on edge', which are described as follows.

1) 'Random': each task randomly selects a time slot $t \in \{1, 2, \ldots, t_{max}\}$, and an offloading decision (locally, edge, cloud).
2) 'Greedy on edge': since the edge computing can usually provide a lower computing delay and relatively low price, each user will offload all tasks to the UAV edge server if it is within the coverage of a UAV. Otherwise, the user decides to wait, process locally, or offload to cloud with certain probabilities. In the simulation, we set the probabilities to 0.8, 0.1, and 0.1, respectively.

Fig. 6 shows the convergence performance of the proposed RL-based computing offloading algorithm. The total cost is calculated by the summation of the cost of each task, which is the weighted sum of delay, energy consumption, and server usage cost. It can be seen that the algorithm converges very fast from the fact that at about 10-th episode the algorithm already converges. The high convergence rate stems from the adopted actor-critic algorithm in which the critic network judges and guides the actor network to learn the policy in each time slot, instead of in each episode for non-actor-critic policy



(a) Average total delay v.s. $\mathcal{C}^e$.



(b) Average total delay v.s. total number of tasks.

Fig. 4. Performance of the proposed VM computing resource allocation and task scheduling algorithm.

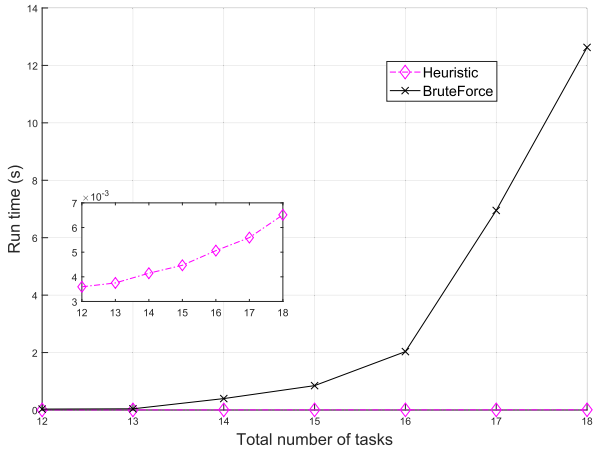


Fig. 5. The run time comparison.

gradient methods. The fast convergence of the algorithm can bring many benefits, such as fast reconfiguration if more users and application are deployed, more flexibility in a dynamic environment, and so forth.

Fig. 7 shows the performance of the proposed computing offloading approach with respect to the UAV server usage

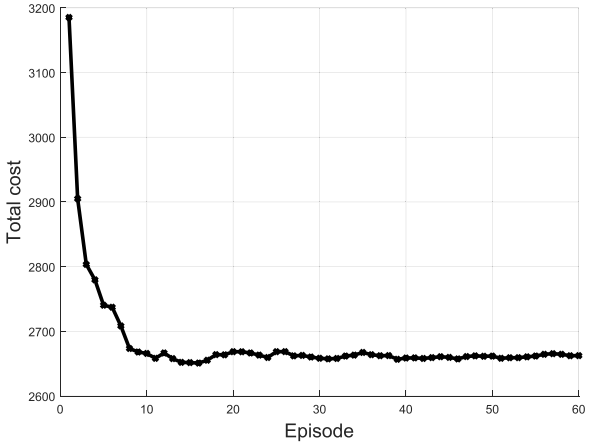


Fig. 6. Convergence performance of our proposed algorithm.

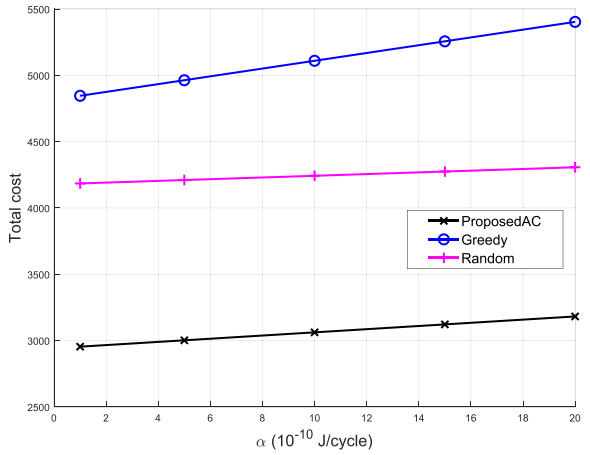


Fig. 7. Total cost v.s. α.

cost weight α. It can be seen that the proposed RL-based approach can achieve the lowest total cost than the other approaches since it can learn the optimal offloading policy through interactions with the environments. 'Greedy' approach suffers the most total cost among the three approaches. This is because 'Greedy' approach forces many tasks content for the UAV channel and edge server computation resources, which increases the times to complete the tasks. In addition, due to the mobility of UAVs, within the time duration in which the task is processing (including the upload, processing, and transmission of the results), the UAV may fly away and the user loses the connection.

In Fig. 8, the main components of the cost, i.e., energy consumption ($E + \mathcal{B} \cdot \alpha$(or β)) and weight delay (ϖT) are shown. It can be seen that the proposed computing offloading approach can achieve the lowest energy consumption and the lowest delay due to the learnt optimal offloading policy. The reason that 'Random' approach achieves the similar total delay as RL-based scheme is that in RL-based scheme, more energy is consumed in transmitting the tasks to the satellite, and in Random scheme, more energy is consumed in locally processing the tasks due to longer local process delay since more tasks are process locally with 'Random' approach (as shown in Fig. 10). However, the 'Greedy' approach has very high

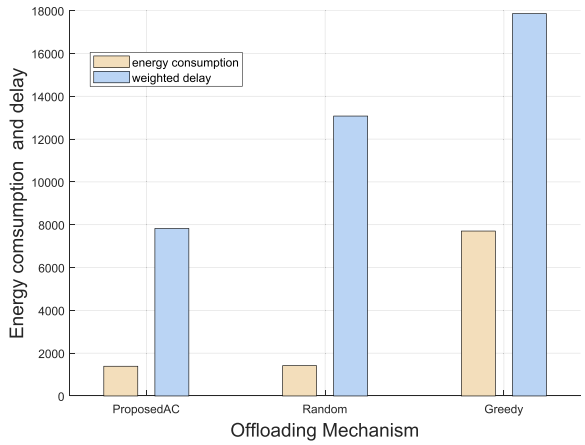


Fig. 8. The energy consumption and weighted delay.

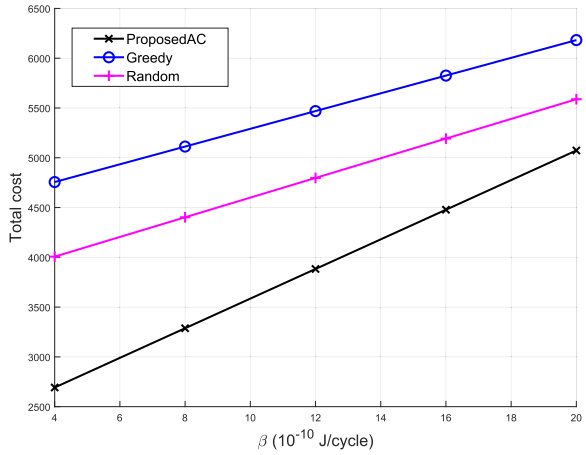


Fig. 9. Total cost v.s. β.

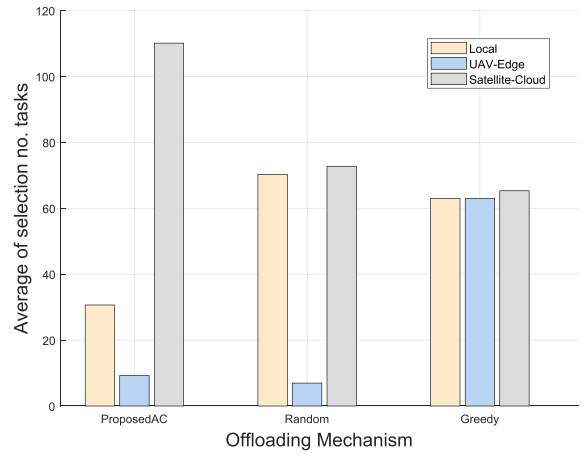


Fig. 10. Offloading means selection.

energy consumption and delay, which is due to that failed execution of tasks in UAV edge servers leads to multiple uploads of the same tasks, and thus it consumes a large amount of energy of the IoT devices and leads to prolonged delay.

Fig. 9 shows the total cost with respect to the cloud server usage cost weight β. Comparing the three approaches, it can be seen the proposed RL-based computing offloading approach

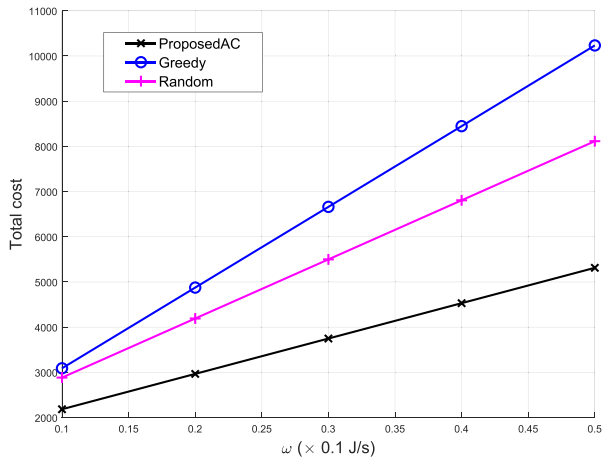


Fig. 11. Total cost v.s. ϖ.

can achieve the lowest average total cost in an episode. The total cost increases with β because the increase of β leads to the increase of $\beta\mathcal{B}^c$ which is a component of the total cost. It can also be seen that the total cost of the proposed approach increases faster the other two approaches, which is because in the current setting of the simulation, the satellite-cloud offloading can achieve relatively better performance than local processing and UAV, if properly chosen. Therefore, the proposed approach learns the environments and chooses cloud offloading with higher probability. This fact can be seen in Fig. 10, which shows the number of selections of each offloading means for each offloading approach. For the proposed approach, it selects satellite-cloud more frequently over the other two offloading means. Compared to satellite-cloud, the local processing results in longer delay due to week local computation capability, while the UAV-edge may suffer the contention problem and high UAV mobility, although it has the benefits of high transmission rate and low server usage cost. The 'Random' and 'Greedy' approaches select almost the same number of local processing and satellite-cloud. The 'Greedy' approach selects more times of UAV-edge since it may wait for the future UAV connection with a high probability if the UAV is current unavailable.

Fig. 11 shows the total cost with respect to the weight on the delay, i.e., ϖ. With the increase of ϖ, the total cost of all the three offloading approach increase, due to the increase of ϖT, which is the delay component of the total cost. However, the proposed offloading approach can achieve the lowest total cost and lower increase rate among the three approaches since it can learn from the environment an optimal policy to reduce the total task delay.

VIII. Conclusion

In this paper, we have investigated the IoT computing offloading problem in SAGIN. We have proposed a joint VM allocation and task scheduling mechanism to efficiently allocate the computing resources to different VMs in the UAV edge server. To offload the computation-intensive tasks, we have proposed an RL-based computing offloading approach to handle the multidimensional SAGIN resources and learns the dynamic network conditions. Deep neural networks, policy

gradient, and actor-critic methods have been employed to improve the learning performance. Simulation results have validated the convergency and efficiency of the proposed approaches. Our work can offer valuable insights to the important yet underexplored field of edge/cloud computing in SAGIN. In the future, we will focus on jointly optimizing the communication, caching, and computing resources in SAGIN.

REFERENCES

[1] M. Shafi *et al.*, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.

[2] N. Cheng *et al.*, "Big data driven vehicular networks," *IEEE Netw.*, vol. 32, no. 6, pp. 160–167, Nov./Dec. 2018.

[3] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.

[4] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81–93, Jan. 2015.

[5] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, to be published.

[6] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[7] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 45–55, Nov. 2014.

[8] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[9] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

[10] F. Lyu *et al.*, "SS-MAC: A novel time slot-sharing MAC for safety messages broadcasting in VANETs," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 3586–3597, Apr. 2018.

[11] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[12] Y. Wu *et al.*, "Secrecy-driven resource management for vehicular computation offloading networks," *IEEE Netw.*, vol. 32, no. 3, pp. 84–91, May/Jun. 2018.

[13] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2714–2741, 4th Quart., 2018.

[14] H. Nishiyama, Y. Tada, N. Kato, N. Yoshimura, M. Toyoshima, and N. Kadowaki, "Toward optimized traffic distribution for efficient network capacity utilization in two-layered satellite networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 3, pp. 1303–1313, Mar. 2013.

[15] Y. Zhou, N. Cheng, N. Lu, and X. Shen, "Multi-UAV-aided networks: Aerial-ground cooperative vehicular networking architecture," *IEEE Veh. Technol. Mag.*, vol. 10, no. 4, pp. 36–44, Dec. 2015.

[16] N. Cheng *et al.*, "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 26–32, Aug. 2018.

[17] Y. Hu and V. O. K. Li, "Satellite-based Internet: A tutorial," *IEEE Commun. Mag.*, vol. 39, no. 3, pp. 154–162, Mar. 2001.

[18] M. Patel *et al.*, "Mobile-edge computing introductory technical white paper," Mobile-edge Comput., Initiative, White Paper, 2014.

[19] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.

[20] T. H. Dinh, D. Niyato, and N. T. Hung, "Optimal energy allocation policy for wireless networks in the sky," in *Proc. IEEE ICC*, Jun. 2015, pp. 3204–3209.

[21] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. Shen, "Software defined space-air-ground integrated vehicular networks: Challenges and solutions," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 101–109, Jul. 2017.

[22] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1046–1061, May 2017.

[23] S. Jagtap, N. Gandhi, and P. Kadam. (2017). *Comparative Study of Project Loon and Facebook Aquila*. [Online]. Available: http://ijesc.org/upload/4d8ea34db8143025dc7aff3880ed9ba9.Comparative%20Study%20of%20Project%20Loon%20&%20Facebook%20Aquila.pdf

[24] W. Quan, Y. Liu, H. Zhang, and S. Yu, "Enhancing crowd collaborations for software defined vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 80–86, Aug. 2017.

[25] W. Shi *et al.*, "Multiple drone-cell deployment analyses and optimization in drone assisted radio access networks," *IEEE Access*, vol. 6, pp. 12518–12529, 2018.

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

[27] 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 *Proc. IEEE ICC*, May 2016, pp. 1–5.

[28] S. Sahni, "Computationally related problems," *SIAM J. Comput.*, vol. 3, no. 4, pp. 262–279, 1974.

[29] P. M. Pardalos and S. A. Vavasis, "Quadratic programming with one negative eigenvalue is NP-hard," *J. Global Optim.*, vol. 1, no. 1, pp. 15–22, 1991.

[30] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, 2016, pp. 1–5.

[31] R. S. Sutton and F. Bach, *Reinforcement Learning—An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[32] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. ICML*, 2014, pp. 1–9.

[33] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992.

[34] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. ICML*, 2016, pp. 1928–1937.

[35] ARM. (2017). *ARM Cortex-M for Beginners*. [Online]. Available: https://community.arm.com/cfs-file/__key/telligent-evolution-components-attachments/01-2142-00-00-00-00-52-96/White-Paper-_2D00_-Cortex_2D00_M-for-Beginners-_2D00_-2016-_2800_final-v3_2900_.pdf

[36] A. Hussain, "Energy consumption of wireless IoT nodes," M.S. thesis, Dept. Inf. Secur. Commun. Technol., Norwegian Univ. Sci. Technol., Trondheim, Norway, NTNU, 2017.

[37] *Study on New Radio (NR) to Support Non Terrestrial Networks*, document Specification # 38.811, 3GPP, 2018. [Online]. Available: http://www.3gpp.org/DynaReport/38811.htm

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

[39] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

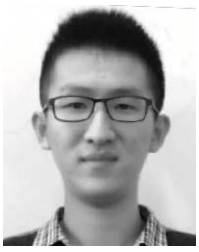
[40] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. HotCloud*, 2010, pp. 4–11.

Nan Cheng (M'16) received the B.E. and M.S. degrees from the Department of Electronics and Information Engineering, Tongji University, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo. He is currently a Joint Professor with the School of Telecommunication, Xidian University. He is also a Joint Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto, and with the Department of Electrical and Computer Engineering, University of Waterloo. His research interests include performance analysis, MAC, opportunistic communication for vehicular networks, unmanned aerial vehicles, and application of artificial intelligence (AI) for wireless networks.

Feng Lyu (M'18) received the B.S. degree in software engineering from Central South University, Changsha, China, in 2013, and the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2018. Since 2018, he has been a Post-Doctoral Fellow with the BBCR Group, Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include vehicular ad hoc networks, cloud/edge computing, and big data driven application design.

Wei Quan (M'14) received the Ph.D. degree in communication and information system from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2014. He is currently an Associate Professor with the School of Electronic and Information Engineering, BJTU. He has published more than 20 papers in prestigious international journals and conferences including *IEEE Communications Magazine*, IEEE WIRELESS COMMUNICATIONS, IEEE NETWORK, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS LETTERS, IFIP Networking, IEEE ICC, and IEEE GLOBECOM. His research interests include key technologies for network analytics, future Internet, 5G networks, and vehicular networks. He is a TPC Member of IEEE ICC in 2017 and 2018, ACM MOBIMEDIA in 2015, 2016, and 2017, and IEEE CCIS in 2015 and 2016. He is also a Member of ACM and a Senior Member of the Chinese Association of Artificial Intelligence (CAAI). He serves as an Associate Editor for the *Journal of Internet Technology* (JIT), *Peer-to-Peer Networking and Applications* (PPNA), and *IET Networks*, and as a technical reviewer for many important international journals.

Conghao Zhou (SM'19) received the B.S. degree from Northeastern University, Shenyang, China, in 2017, and the M.S. degree from the University of Illinois at Chicago, Chicago, IL, USA, in 2018. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include space-air-ground integration networks and machine learning in wireless networks.



Hongli He received the B.Sc. degree in information engineering from Zhejiang University, Hangzhou, China, in 2014, where he is currently pursuing the Ph.D. degree with the Institute of Information and Communication Engineering. His current research interests include video streaming in vehicular ad hoc networks, local thermal equilibrium in unlicensed spectrum, and edge cloud computing.

Weisen Shi (SM'15) received the B.S. degree from Tianjin University, Tianjin, China, in 2013, and the M.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2016. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His interests include drone communication and networking, network function virtualization, and vehicular networks.

Xuemin (Sherman) Shen (M'97–SM'02–F'09) received the B.Sc. degree in electrical engineering from Dalian Maritime University, China, in 1982, and the M.Sc. and Ph.D. degrees in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1987 and 1990, respectively. He is currently a University Professor and an Associate Chair for Graduate Studies with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on resource management, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He is a Registered Professional Engineer of ON, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society. He was an Elected Member of the IEEE ComSoc Board of Governor and the Chair of the Distinguished Lecturers Selection Committee. He was a recipient of the Excellent Graduate Supervision Award in 2006. He received the Premiers Research Excellence Award (PREA) from the Province of Ontario, Canada, in 2003. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom16, Infocom14, IEEE VTC10 Fall, and Globecom07, the Symposia Chair for IEEE ICC10, the Tutorial Chair for IEEE VTC11 Spring and IEEE ICC08, the General Co-Chair for ACM Mobihoc15, Chinacom07, and QShine06, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications and P2P Communications and Networking. He also serves/served as the Editor-in-Chief for the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, *Peer-to-Peer Networking and Application*, and *IET Communications*; a Founding Area Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS; an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, *Computer Networks*, and *ACM/Wireless Networks*; and the Guest Editor for IEEE JSAC, IEEE WIRELESS COMMUNICATIONS, *IEEE Communications Magazine*, and *ACM Mobile Networks and Applications*.