# Adaptive Resource Allocation for IoT with Computing Power Network Based on RIS-UAV-Aided NOMA-THz Communication

Kaiwen Pan, Meng Li, *Senior Member, IEEE*, Suyu Lv, *Member, IEEE*, Pengbo Si, *Senior Member, IEEE*, Haijun Zhang, *Fellow, IEEE*, and F. Richard Yu, *Fellow, IEEE*

***Abstract*—The integration of advanced technologies such as sixth-generation mobile communications (6G), artificial intelligence (AI) and blockchain has given new impetus to the development of the Internet of Things (IoT). However, these applications require higher computational power and lower latency, which present challenges to traditional network architectures. To address these issues, this paper proposes a novel computing power network (CPN) architecture based on reconfigurable intelligent surface (RIS)-unmanned aerial vehicle (UAV)-assisted non-orthogonal multiple access (NOMA)-Terahertz (THz) communication, and it aims to meet high computational demands. In the proposed scheme, CPN is introduced to assist IoT devices in executing tasks, thereby enhancing data processing. Concretely, THz communications and NOMA technologies are utilized to increase data rates and spectral efficiency. The combination of RIS and UAV shows promise in overcoming the challenges of high path loss and high sensitivity to blockage in THz communication, thereby improving system performance. To increase the efficiency of the proposed architecture, it is crucial to rationally allocate computational and transmission resources. Therefore, a joint optimization problem is formulated to minimize system consumption, encompassing both time and energy usage. To achieve efficient resource allocation, an adaptive N-Step method based on soft actor-critic (SAC) algorithm is employed. Simulation results demonstrate the superiority of the proposed method over the existing baselines.**

***Index Terms*—Computing power networks, Internet of Things, Terahertz communication, reconfigurable intelligent surface, deep reinforcement learning.**

## I. Introduction

In recent years, the number of Internet of Things (IoT) devices has exhibited significant growth [1]. This rapid expansion reflects the potential market requirements for IoT technology and its growing importance in our daily life. Concurrently, the emergence of advanced technologies, such as sixth-generation mobile communications (6G), artificial intelligence (AI), and blockchain, has injected new momentum into the development of IoT [2]. The integration of these advanced technologies with IoT has spawned numerous innovative applications, resulting in profound impacts and transformative changes across various industries. For instance, the integration of AI technology enables IoT devices to perform more intelligent data processing and decision-making [3]; the implementation of blockchain technology enhances the security and transparency of IoT systems [4]; the high-speed and low-latency performance of the 6G network further improves the connectivity and responsiveness of IoT devices, facilitating the realization of more complex and diverse applications [5], [6]. The integration of these technologies not only enhances the performance of IoT systems but also creates innovative application scenarios such as smart cities, remote medical care, industrial IoT, etc., indicating broad prospects for future development [2], [6].

However, these applications demand higher computational power to handle large volumes of real-time data, posing challenges to meeting quality of service (QoS) requirements. Traditional network architectures struggle to meet these high computational demands due to limited capacity and scalability issues. To address these challenges, a new networking architecture, known as computing power networks (CPN), has been proposed [7], [8]. CPN connects different types of computing resources through networking, allowing them to collaborate and to be utilized flexibly. It encompasses various computing architectures, representing a multilayer network that integrates cloud devices, edge computing nodes (ECN), and client equipment, among other elements [7].

The wireless channel conditions in the CPN are crucial and are often affected by environmental factors, such as obstacles and interference from other devices. To better support communication in the system, employing a higher carrier frequency is effective. Terahertz (THz) technology is seen as a promising approach to advance next-generation wireless communication systems [9]. In the THz communication-enabled CPN, the network is expected to accommodate a considerable number of data transmissions simultaneously. However, conventional orthogonal multiple access (OMA) technologies may not always efficiently support data transfer within the system. As a solution, non-orthogonal multiple access (NOMA) is gaining wider adoption. By allowing more nodes to reuse a single subchannel, it enhances spectral efficiency, thereby increasing resource utilization in THz networks [10].

Although NOMA technology is able to enhance the efficiency of THz communications, there are still a number of challenges to overcome as THz communication technologies have high path loss and sensitivity to blockages. Fortunately, the reconfigurable intelligent surface (RIS) can solve the problem of path loss and blockage in THz communication. The RIS consists of lots of reflecting elements, and each element is able to change its phase shift independently, allowing for the adjustment of the phase of the incident signal [11] [12]. This method establishes a new cascade transmission channel to re-

duce the impact of blockage on the direct transmission channel [13]. Previous studies have often assumed a fixed position for the RIS, which may not be suitable for THz communication systems due to their shorter transmission distance and higher sensitivity to obstacles [14]. Therefore, adjusting the position of the RIS based on the location of client nodes is necessary to enhance system performance. Meanwhile, with the rapid advancement of unmanned aerial vehicle (UAV) technology, equipping a RIS on an UAV has become an avenue that is worth exploring further [15].

In this paper, to enhance the efficiency of the proposed CPN architecture, it is necessary to allocate both the computational and transmission resources. To minimize the system's overall consumption, we establish a joint optimization problem that simultaneously optimizes the task offloading decision, transmission power allocation, computation capacity allocation, RIS reflection coefficients, and UAV trajectory. Due to the non-convexity of the optimization problem, traditional optimization methods may not be suitable. Thus, we introduce an adaptive N-Step method based on soft actor-critic (SAC) algorithm to obtain a near-optimal solution efficiently. Additionally, this method has been compared with existing baselines, and simulation results indicate that it outperforms these baselines. Compared with other related works, this paper has made improvements in system architecture, optimization objectives and algorithms, achieving favorable results and demonstrating significant disparities and advantages.

The main contributions of this paper are listed as follows.

- To address the increased computational demands of processing large volumes of real-time data for emerging IoT applications, we design and propose a new CPN architecture that integrates RIS-UAV and NOMA-THz communication technologies.
- To reduce the time and energy consumption of the proposed CPN system, we formulate an optimization problem aimed at minimizing total consumption through the joint optimization of transmission power allocation, computation capacity allocation, task offloading decisions, RIS reflection coefficients, and UAV trajectory.
- Due to the non-convexity of the optimization problem, traditional optimization methods may not be suitable for solving it. Deep reinforcement learning (DRL) is regarded as a promising approach. After formulating the Markov decision process (MDP) model, we introduce an adaptive N-Step method based on the SAC algorithm to efficiently obtain a near-optimal solution that meets the application requirements.
- The proposed method is evaluated against existing baseline methods across various scenarios, with simulation results showing its superior performance.

The remainder of this paper is structured as follows. Section II reviews the relevant research. Section III provides a detailed description of the system model of the proposed CPN and defines the optimization problem. Section IV develops the MDP model for the optimization problem and introduces an adaptive N-Step SAC method to solve it. The simulation results are discussed and analyzed in Section V. Finally, the paper concludes in Section VI.

## II. Related Work

In this section, we review the relevant literatures on computation offloading in CPN, RIS-UAV-aided wireless communication, THz communication, and NOMA technology.

### A. Computation Offloading in CPN

In CPN, task offloading enhances mobile device performance by transferring computationally intensive tasks to the ECN, thus improving overall system performance.

Minimizing the system's overall latency is the primary goal of current researches about computation offloading. Alternatively, some approaches consider the time and energy cost simultaneously, aiming to look for an appropriate trade-off [16]. Liu *et al.* [17] introduced a one-dimensional search method to optimize client node offloading decisions. The offloading decision of each client node is determined by a computation offloading policy module, and three baseline methods are compared with the method to confirm its superior performance: local execution, cloud execution, and greedy policy. Their one-dimensional search method has a lower average execution time of tasks than the baseline methods.

### B. RIS-UAV-Aided Wireless Communication

A RIS is composed of lots of reflecting elements that can vary the exit phase and angle of signal by adjusting the phase of incoming signal [13], thereby improving signal strength to the target and reducing signal strength to other objects.

The integration of RIS with UAV has the potential to significantly enhance the communication efficiency of the system. Various studies have demonstrated that this approach can lead to improvements in computation rate, transmission rate, and network capacity [18]–[20]. These improvements can be realized through the optimization of RIS-UAV trajectories, phase shifts, and other system parameters. Truong *et al.* [18] utilized a RIS-UAV system to support transmission. To enhance the system's overall transmission rate, they employed a DRL method to optimize the RIS-UAV parameters. In [19], Liu *et al.* used a RIS-mounted UAV to improve the system's transmission performance. The problem was divided into three subproblems, and an alternating iterative method was employed to obtain suboptimal solutions. Zhang *et al.* [20] presented a RIS-UAV-aided NOMA system with the objective of enhancing network capacity through the joint optimization of RIS phase shifts and UAV trajectory, utilizing the double deep Q-network algorithm to solve the problem.

### C. THz Communication

THz communication has great potential for future communication systems, utilizing high-frequency bands ranging from $10^{11}$ Hz to $10^{13}$ Hz [21]. As the demand for high data rates increases in 6G and beyond wireless communication, THz communication is anticipated to meet the high transmission requirements of users and is envisioned as a key technology [21].

In [22], a deep learning method was proposed to achieve channel estimation in RIS-assisted THz communication. Using RIS, the problems of significant path loss and blocking effects in THz communication can be addressed, thereby improving the performance of THz communication [23]. Wang *et al.* [24] studied an UAV-aided THz-band communication edge computing system. The objective is to reduce system latency by optimizing the decisions within the THz communication system. They proposed a DRL-based method and demonstrated its efficiency.

### D. Optimization of Power-Domain NOMA

NOMA is widely used in wireless communication due to its reliability, spectral efficiency, and support for high-capacity connections. NOMA allocates the same time-frequency resources to different devices, and these devices are non-orthogonal to each other, which differs from traditional OMA methods. In power-domain NOMA, the transmission power of users depends on the channel conditions.

Power-domain NOMA has become one of the most important multiple access techniques. Meanwhile, the optimization problem in NOMA systems has received extensive attention from researchers. Aiming to enhance the overall transmission speed of the multi-user NOMA system, Rezwan *et al.* [25] used a DRL method to allocate the transmission power among users and proposed a user clustering algorithm to obtain an efficient clustering scheme. In [26], Qian *et al.* studied a cellular edge computing system where users transmit their data to the edge server using NOMA. They proposed an efficient layered algorithm and demonstrated its efficiency through simulations comparing it with baseline methods. A NOMA edge computing system with a RIS that simultaneously reflects and transmits was studied in [27]. They introduced a DRL method for the allocation of resources and offloading decisions, confirming its efficiency through simulations.

TABLE I: SUMMARY OF NOTATIONS

| Symbol | Definition |
|---|---|
| $\boldsymbol{\mathcal{T}}$ | The set of all time slots |
| $\boldsymbol{\mathcal{N}}$ | The set of all client nodes |
| $\boldsymbol{\mathcal{K}}$ | The set of reflecting elements of RIS |
| $\boldsymbol{q_{u,i}}$ | The coordinates of client node $i$ |
| $\boldsymbol{q_r^t}$ | The coordinates of RIS-UAV |
| $\boldsymbol{q_b}$ | The coordinates of BS antenna |
| $\boldsymbol{v_r^t}$ | The movement vector of the RIS-UAV |
| $v_r^{max}$ | Maximum movement distance of RIS-UAV |
| $D_{cb,i}^t$ | The distance from BS to client node $i$ |
| $D_{cr,i}^t$ | The distance from client node $i$ to RIS-UAV |
| $D_{rb}^t$ | The distance from RIS-UAV to BS |
| $P_i^t$ | The transmit power of client node $i$ |
| $\boldsymbol{\Theta_t}$ | The phase shift matrix of RIS |
| $h_{cb,i}^t$ | Direct uplink channel gain |
| $h_{crb,i}^t$ | Cascade uplink channel gain |
| $R_{cb,i}^t$ | The transmission rate from client node $i$ to BS |
| $\boldsymbol{M_i^t}$ | The computation task of client node $i$ |
| $\phi_i^t$ | The maximum tolerable delay of task $\boldsymbol{M_i^t}$ |
| $d_i^t$ | The size of data needed for computing $\boldsymbol{M_i^t}$ |
| $\kappa_i^t$ | The CPU cycles required to accomplish $\boldsymbol{M_i^t}$ |
| $C_i$ | The computation capacity of client node $i$ |
| $C_{ECN}^{total}$ | The total computation capacity of the ECN |
| $C_{ECN}^t$ | The available computation capacity of the ECN |
| $\tau_i^t$ | The remaining execution time of tasks on client node $i$ |
| $P_i^{idle}$ | The power consumption of client node $i$ in idle state |
| $\boldsymbol{\eta_t}$ | Computation capacity allocation vector |
| $\boldsymbol{\alpha_t}$ | Task offloading decision vector |
| $T_{l,i}^t$ | Time consumption for local computing of $\boldsymbol{M_i^t}$ |
| $T_{o,i}^t$ | Time consumption for offloading $\boldsymbol{M_i^t}$ |
| $E_{l,i}^t$ | Energy consumption for local computing of $\boldsymbol{M_i^t}$ |
| $E_{o,i}^t$ | Energy consumption for offloading $\boldsymbol{M_i^t}$ |
| $O_{l,i}^t$ | Total consumption for local computing $\boldsymbol{M_i^t}$ |
| $O_{o,i}^t$ | Total consumption for offloading $\boldsymbol{M_i^t}$ |
| $O_i^t$ | Total consumption for $\boldsymbol{M_i^t}$ |
| $\lambda$ | The carrier wavelength of the used THz band |
| $\zeta$ | The medium absorption factor |
| $B$ | The bandwidth of the used THz band |
| $\omega$ | The weight of the RIS-UAV |
| $\iota$ | The preference for latency |
| $\varsigma$ | The energy efficiency parameter |
| $G_{re}$ | The antenna gain of the receiver |
| $G_{tr}$ | The antenna gain of the transmitter |

## III. System Model and Problem Formulation

In this section, we present the system model of the proposed CPN based on RIS-UAV-aided NOMA-THz communication, and define the optimization problem. The system model is illustrated in Fig. 1.

### A. Network Model

*1) BS Plane:* The base station (BS) serves as the central node of the system, equipped with an ECN to provide computational resources for client nodes within the system. The total computation capacity of the ECN is $C_{ECN}^{Total}$. The height of the transmission antenna of the BS is $h_b$, thus the coordinates of BS antenna are $\boldsymbol{q_b} = \{0, 0, h_b\}$.

*2) Client Node Plane:* There are $N$ client nodes in the system, indexed by $\boldsymbol{\mathcal{N}} = \{1, 2, \ldots, N\}$. The coordinate of client node $i$ is fixed and could be denoted by $\boldsymbol{q_{u,i}} = \{x_{u,i}, y_{u,i}, 0\}$. The computation capacity of client node $i$ is $C_i$.

*3) RIS-UAV Plane:* The UAV is equipped with a RIS; therefore, the RIS-UAV can adjust the phase of the reflecting elements and change its position to improve transmission performance. We use $\boldsymbol{\mathcal{K}} = \{1, 2, \ldots, K\}$ to express $K$ reflecting elements of RIS. In time slot $t$, $\boldsymbol{\Theta} = diag(\Theta_1, \Theta_2, \ldots, \Theta_K)$ denotes the phase shift matrix, $\Theta_k = \beta_k e^{j\theta_k}$, where $\theta_k$ denotes the phase shift, $\beta_k$ denotes the amplitude of $k$-th reflecting element of the RIS. This paper assumes that $\beta_k = 1,\ \forall k \in \boldsymbol{\mathcal{K}}$. The position of RIS-UAV is $\boldsymbol{q_r^t} = \{x_r^t, y_r^t, h_r^t\}$. The weight of RIS-UAV is $\omega$.

*4) Task Plane:* Our assumption is that at the start of time slot $t$, each of the client nodes may have a computationally intensive task to do. We use $\boldsymbol{M_i^t} = \{d_i^t, \kappa_i^t, \phi_i^t\}$ to denote the information of the task of client node $i$, where $d_i^t$ is the size of data needed for computing $\boldsymbol{M_i^t}$, $\kappa_i^t$ is the CPU cycles required to execute $\boldsymbol{M_i^t}$, $\phi_i^t$ is the maximum tolerable delay of task $\boldsymbol{M_i^t}$.

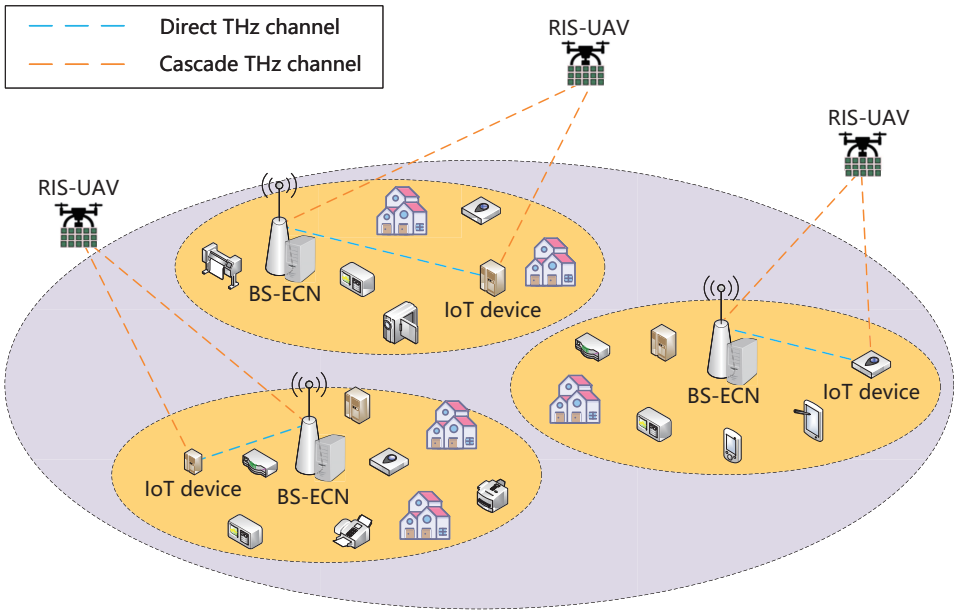


Fig. 1. System model of the proposed CPN.

### B. Communication Model

This section introduces the communication model of the proposed CPN architecture. In the proposed CPN, the transmission links are composed of direct links and cascade links. The direct links are between client nodes and BS, and the cascade links are between client nodes, RIS-UAV, and BS. Then, the NOMA technology is used to enhance the transmission efficiency.

*1) Direct THz Transmission Links:* The coordinates of the BS antenna and the client node $i$ are $\boldsymbol{q_b} = \{0, 0, h_b\}$ and $\boldsymbol{q_{u,i}} = \{x_{u,i}, y_{u,i}, 0\}$ respectively. The distance between client node $i$ and BS antenna is denoted by $D^t_{cb,i}$. Then $D^t_{cb,i}$ can be represented as

$$D^t_{cb,i} = \sqrt{{x_{u,i}}^2 + {y_{u,i}}^2 + h_b^2}. \tag{1}$$

As outlined in [28] Section II.A, the direct THz channel gain between client node $i$ and BS in $t$ is denoted by $h^t_{cb,i}$ and can be calculated as

$$h^t_{cb,i} = \frac{\lambda}{4\pi D^t_{cb,i}} \exp(-[\frac{j2\pi D^t_{cb,i}}{\lambda} + \frac{\zeta D^t_{cb,i}}{2}]), \tag{2}$$

where $\lambda$ denotes the carrier wavelength and $\zeta$ is the medium absorption factor for the THz band.

*2) RIS-Assisted Cascade THz Transmission Links:* The RIS-UAV's coordinates in $t$ is defined as $\boldsymbol{q_r^t} = \{x_r^t, y_r^t, h_r^t\}$. The separation between client node $i$ and the RIS-UAV is denoted as $D^t_{cr,i}$ and can be represented as

$$D^t_{cr,i} = \sqrt{(x^t_{u,i} - x^t_r)^2 + (y^t_{u,i} - y^t_r)^2 + {h^t_r}^2}, \tag{3}$$

the separation between the RIS-UAV and the BS antenna is denoted as $D^t_{rb}$ and can be calculated by

$$D^t_{rb} = \sqrt{{x^t_r}^2 + {y^t_r}^2 + (h^t_r - h_b)^2}. \tag{4}$$

According to [28] Section II.B, the cascade channel gain in THz transmission scenario can be obtained by

$$\begin{aligned} h^t_{crb,i} = &\exp(-\frac{j2\pi(D^t_{cr,i} + D^t_{rb})}{\lambda} - \frac{\zeta(D^t_{cr,i} + D^t_{rb})}{2}) \\ &(\frac{\lambda}{8\sqrt{\pi^3} D^t_{cr,i} D^t_{rb}})(\boldsymbol{a^t_{cr,i}})^{\boldsymbol{T}} \boldsymbol{\Theta_t} \boldsymbol{a^t_{rb}}, \end{aligned} \tag{5}$$

where $\boldsymbol{a^t_{cr,i}}$ represents the RIS's receive array response from client node $i$ to RIS, $\boldsymbol{a^t_{rb}}$ represents the RIS's transmit array response from RIS to BS.

*3) Transmission Rate In NOMA Scenario:* If client node $i$ decides to upload its data to the BS to execute its task on the ECN, i.e. $\alpha^t_i = 1$, the received signal of BS is determined by direct channel conditions, cascade channel conditions, transmit power, and noise.

$$y^t_i = (h^t_{cb,i} + h^t_{crb,i})\sqrt{P^t_i} s^t_i + N^t_i, \tag{6}$$

where $s^t_i$ denotes the signal from client node $i$, $P^t_i$ is client node's transmission power. $N^t_i$ is composed of signals transmitted by other client nodes and channel noise, thus it can be expressed as

$$N^t_i = \sum_{j \in \boldsymbol{\mathcal{N}},\, j \neq i} \alpha^t_j (h^t_{cb,j} + h^t_{crb,j})\sqrt{P^t_j} s^t_j + n^t_i, \tag{7}$$

where $n^t_i$ is the additive white Gaussian noise with variance $\sigma^{t2}_i$.

We use successive interference cancellation (SIC) to eliminate interference caused by client nodes with stronger channel gains in the uplink NOMA scenario. Thus, for a signal, we only consider the effect of other signals whose channel conditions are not as good as it. Then the noise experienced by client node $i$, denoted as $N^{t\prime}_i$, can be represented as

$$N^{t\prime}_i = \sum_{j \in \boldsymbol{\mathcal{N}},\, j \neq i} \alpha^t_j \mu^t_{i,j} (h^t_{cb,j} + h^t_{crb,j})\sqrt{P^t_j} s^t_j + n^t_i, \tag{8}$$

where $\mu_{i,j}^t$ is used to determine whether the signal transmitted by client node $j$ should be counted in the noise of client node $i$ or not. If the channel conditions of client node $j$ are better than those of client node $i$, the noise generated by this node can be eliminated by the SIC for client node $i$ due to its priority to be decoded and therefore is not counted as part of the noise of client node $i$. Otherwise, as the signal has not been decoded yet, it needs to be counted as part of the noise of client node $i$. Thus, $\mu_{i,j}^t$ can be defined as

$$\mu_{i,j}^t = \begin{cases} 1, & if\ |h_{cb,j}^t + h_{crb,j}^t|^2 \leq |h_{cb,i}^t + h_{crb,i}^t|^2, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Then, referring to the Eq.(3) in [14] the signal-to-interference-plus-noise-ratio (SINR) of client node $i$ in time slot $t$ in NOMA-THz communication network, denoted as $SINR_i^t$, is given by

$$SINR_i^t = \frac{P_i^t |h_{cb,i}^t + h_{crb,i}^t|^2}{\sum_{j=1,j\neq i}^{N} \alpha_j^t \mu_{i,j}^t P_j^t |h_{cb,j}^t + h_{crb,j}^t|^2 + {\sigma_i^t}^2 B}, \tag{10}$$

where $B$ is the bandwidth of the used THz-band.

Hence, the uplink transmission rate from client node $i$ to the BS can be denoted as $R_{cb,i}^t$ and calculated by the Shannon formula using the SINR in Eq.(10). Thus, the transmission rate can be expressed as

$$R_{cb,i}^t = B\log_2(1+\frac{G_{re}G_{tr}P_i^t|h_{cb,i}^t + h_{crb,i}^t|^2}{\sum_{j=1,j\neq i}^{N} \alpha_j^t \mu_{i,j}^t P_j^t |h_{cb,j}^t + h_{crb,j}^t|^2 + {\sigma_i^t}^2 B}), \tag{11}$$

where $G_{re}$ is the antenna gain of the receiver and $G_{tr}$ is the antenna gain of the transmitter.

### C. Computing Model

*1) Local Computing Model:* Let the time consumption for locally executing task $\boldsymbol{M_i^t}$ be $T_{l,i}^t$, $E_{l,i}^t$ is the energy cost of executing $\boldsymbol{M_i^t}$. Then, $T_{l,i}^t$ is the sum of the calculation time and the remaining task execution time on client node $i$ when time slot $t$ starts, which can be represented as

$$T_{l,i}^t = \frac{\kappa_i^t}{c_i} + \tau_i^t, \tag{12}$$

where $\tau_i^t$ denotes the remaining execution time of tasks on client node $i$ when time slot $t$ starts.

According to [29], the energy consumption for locally executing task $\boldsymbol{M_i^t}$ is associated with the hardware of client node $i$. Thus, the energy consumption $E_{l,i}^t$ can be calculated as

$$E_{l,i}^t = \varsigma_i c_i^2 \kappa_i^t, \tag{13}$$

where $\varsigma_i$ represents the energy utilisation of client node $i$, related to its hardware.

Thus, the total consumption for local computing $\boldsymbol{M_i^t}$ is denoted as $O_{l,i}^t$ and can be calculated using Eq.(12) and Eq.(13)

$$O_{l,i}^t = \iota T_{l,i}^t + (1-\iota)E_{l,i}^t, \tag{14}$$

where $\iota \in [0,1]$ is the preference of the latency. A higher value of $\iota$ indicates a stronger preference for minimizing latency in the system.

*2) Offloading Computing Model:* Client nodes can offload their tasks to the ECN via the communication network mentioned above. The task offloading process can be summarised in three steps.

*a)* The client node transmits the data of task $\boldsymbol{M_i^t}$ to the ECN.

*b)* The ECN executes the task $\boldsymbol{M_i^t}$.

*c)* The ECN returns the execution result to the client node.

If client node $i$ uploads the data required for the task $\boldsymbol{M_i^t}$, the time consumption $T_{o,i1}^t$ is given by

$$T_{o,i1}^t = \frac{d_i^t}{R_{cb,i}^t}, \tag{15}$$

where $d_i^t$ is the size of data required for computing $\boldsymbol{M_i^t}$, $R_{cb,i}^t$ is the uplink transmission rate from client node $i$ to BS given by Eq.(11).

The energy consumption $E_{o,i1}^t$ is

$$E_{o,i1}^t = T_{o,i1}^t P_i^t, \tag{16}$$

where $P_i^t$ is the transmission power of client node $i$.

The time consumption $T_{o,i2}^t$ of executing the task at the edge computing node can be expressed as

$$T_{o,i2}^t = \frac{\kappa_i^t}{\eta_i^t}. \tag{17}$$

The energy consumption $E_{o,i2}^t$ of executing $\boldsymbol{M_i^t}$ in ECN consists of execution consumption in the ECN and the client node's idle state consumption. It can be calculated as

$$E_{o,i2}^t = \varsigma_m {\eta_i^t}^2 \kappa_i^t + T_{o,i2}^t P_i^{idle}, \tag{18}$$

where $P_i^{idle}$ is the power consumption of client node $i$ in idle state, $\varsigma_m$ represents the energy efficiency parameter depending on the hardware of ECN.

While the ECN accomplishes the task, the execution result needs to be sent to the client node, the time consumption $T_{o,i3}^t$ and the energy consumption $E_{o,i3}^t$ are given by

$$T_{o,i3}^t = \frac{b_i^t}{R_{b,i}^t}, \tag{19}$$

and

$$E_{o,i3}^t = T_{o,i3}^t P_{b,i}^t + T_{o,i3}^t P_i^{idle}, \tag{20}$$

where $b_i^t$ is the execution result size, $P_{b,i}^t$ is the transmitting power that base station transmits execution result to client node $i$, $R_{b,i}^t$ is download rate.

The overall time consumption of the task is $T_{o,i}^t$ and energy consumption is $E_{o,i}^t$

$$T_{o,i}^t = T_{o,i1}^t + T_{o,i2}^t + T_{o,i3}^t, \tag{21}$$

and

$$E_{o,i}^t = E_{o,i1}^t + E_{o,i2}^t + E_{o,i3}^t. \tag{22}$$

According to [30], the downlink speed is typically very high, and execution results are sufficiently small to be considered negligible in general. Therefore, the consumption of returning the execution result to the client node is neglected in the proposed system model.

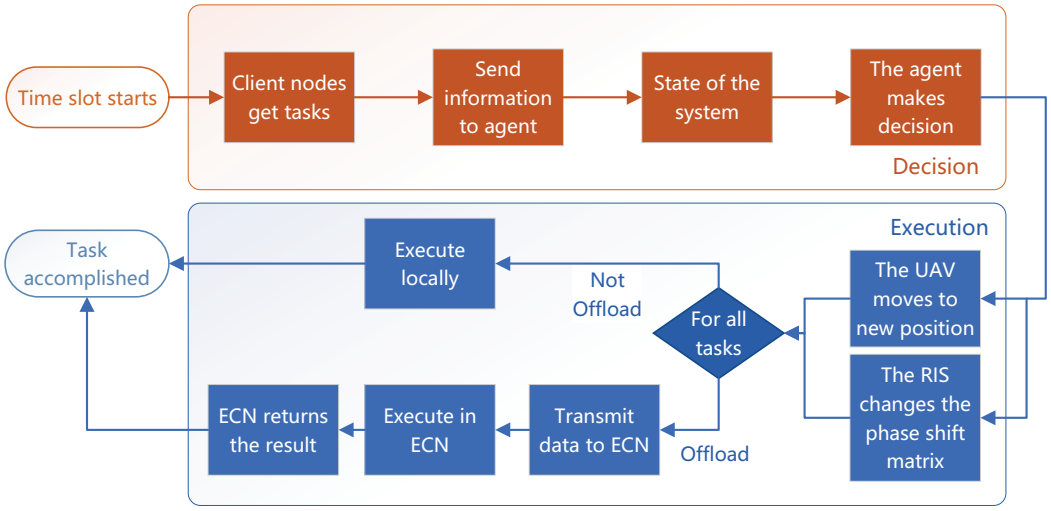


Fig. 2. System workflow of the proposed CPN.

The total consumption for offloading $M_i^t$ is denoted as $O_{o,i}^t$ and can be calculated based on Eq.(21) and Eq.(22)

$$O_{o,i}^t = \iota T_{o,i}^t + (1-\iota) E_{o,i}^t. \tag{23}$$

The system workflow for the completion of the task is illustrated in Fig. 2.

### D. UAV Model

*1) UAV Motion:* The UAV motion vector in time slot $t$ is $\boldsymbol{v_r^t} = \{\Delta x_r^t, \Delta y_r^t, \Delta h_r^t\}$, the new coordinate $\boldsymbol{q_r^t}$ of the RIS-UAV after moving is expressed as

$$\boldsymbol{q_r^t} = \boldsymbol{q_r^{t-1}} + \boldsymbol{v_r^t} \min(1, \frac{v_r^{max}}{|\boldsymbol{v_r^t}|}), \tag{24}$$

where $v_r^{max}$ represents the maximum length the UAV can move within a time slot, which is related with the maximum movement speed of UAV.

*2) UAV Energy Consumption:* According to [31], the energy consumption of UAV during $t$ is given by

$$E_u^t = \frac{\omega}{2} \left|\left| \frac{\boldsymbol{q_r^t} - \boldsymbol{q_r^{t-1}}}{\sqrt{t_m}} \right|\right|^2, \tag{25}$$

where $t_m$ is the maximum flight time of UAV, $\omega$ is the weight of UAV, which is related to its hardware.

### E. Problem Formulation

Based on the above analysis, the system's overall consumption in time slot $t$ can be expressed as

$$O_t = \sum_{i=1}^{N} [\alpha_i^t O_{o,i}^t + (1-\alpha_i^t) O_{l,i}^t] + (1-\iota) E_u^t, \tag{26}$$

where $E_u^t$ is the energy cost of the UAV, $\alpha_i^t$ is the offloading decision of client node $i$, $O_{o,i}^t$ is the total consumption associated with offloading $\boldsymbol{M_i^t}$, $O_{l,i}^t$ is the total consumption of local computing $\boldsymbol{M_i^t}$.

The problem is then formulated based on aforementioned system model, which is derived from the equations mentioned above.

$$\begin{aligned}
&\min_{\boldsymbol{\alpha_t}, \boldsymbol{\eta_t}, \boldsymbol{\Phi_t}, \boldsymbol{v_r^t}, \boldsymbol{P_t}} \quad \sum_{t \in \boldsymbol{\mathcal{T}}} O_t, \\
s.t. \quad & C1: \alpha_i^t \in \{0,1\}, \forall i \in \{1,2,\ldots,N\}, \\
& C2: P_i^t \le P_{max}, \forall i \in \{1,2,\ldots,N\}, \\
& C3: \sum_{i=1}^{N} \alpha_i^t \eta_i^t \le C_{ECN}^t, \\
& C4: \alpha_i^t T_{o,i}^t + (1-\alpha_i^t) T_{l,i}^t \le \phi_i^t, \forall i \in \{1,2,\ldots,N\}, \\
& C5: |\boldsymbol{v_r^t}| \le v_r^{max}, \\
& C6: \theta_k^t \in [0, 2\pi], \ \forall k \in \boldsymbol{\mathcal{K}}.
\end{aligned} \tag{27}$$

where $\boldsymbol{\alpha_t} = \{\alpha_1^t, \alpha_2^t, \ldots, \alpha_N^t\}$ is the task offloading decision vector, i.e. if $\alpha_2^t = 1$ then the task $M_i^t$ will be offload, and if $\alpha_2^t = 0$ then the task $M_i^t$ will be execute locally. $\boldsymbol{\eta_t} = \{\eta_1^t, \eta_2^t, \ldots, \eta_N^t\}$ is the computation capacity allocation vector, $\eta_i^t$ is computation capacity allocated by ECN to execute the task $M_i^t$. $\boldsymbol{\Phi_t} = \{\theta_1^t, \theta_2^t, \ldots, \theta_K^t\}$ is the phase shift vector of RIS. $\boldsymbol{v_r^t} = \{\Delta x_r^t, \Delta y_r^t, \Delta h_r^t\}$ is the movement vector of the RIS-UAV. $\boldsymbol{P_t} = \{P_1^t, P_2^t, ..., P_N^t\}$ is the transmit power of client node $i$.

In Eq.(27), $C1$ represents that the client node $i$ could only choose one way to accomplish the task $M_i^t$, either offloading computation or local computation. $C2$ guarantees that the client node's transmission power does not exceed the hardware limit. $C3$ indicates that the computation capacity that the ECN allocated to offloading tasks doesn't surpass the available computational capacity of the edge computing node in time slot $t$. $C4$ guarantees that the total time required to complete a task should not extend beyond $\phi_i^t$. $C5$ ensures that the distance moved by the RIS-UAV does not exceed the maximum limit $C6$ guarantees that the phase shift of the RIS remains within the range of $[0, 2\pi]$.

## IV. Proposed Solutions

In this section, the proposed problem is formulated as an MDP. Then an adaptive N-Step decision making method based on SAC algorithm is adopted in order to solve the formulated MDP problem.

### A. Markov Decision Process Formulation

Based on the aforementioned system model, it can be obtained that the system discussed in this paper satisfies the Markov decision property, and the conditional probability distribution of the future state of the system depends on the current state and the actions taken by the agent in the current state.

$$p\left(s_{t+1} \mid s_t, a_t\right) = p\left(s_{t+1} \mid s_1, s_2, s_3, \ldots, s_t, a_t\right). \tag{28}$$

In this section, the problem in Eq.(27) is formulated as an MDP.

*1) State Space:* When a time slot begins, each client node and the edge computing node transmit their information to the CPN controller. Then the intelligent agent of the controller aggregates this information as the state of the system.

$$\boldsymbol{s(t)} = \{\boldsymbol{\mathcal{M}_t}, \boldsymbol{q_r^{t-1}}, \boldsymbol{\tau_t}, C_{ECN}^t\}, \tag{29}$$

where $\boldsymbol{\mathcal{M}_t} = \{\boldsymbol{M_1^t}, \boldsymbol{M_2^t}, ..., \boldsymbol{M_N^t}\}$ is the task information matrix in time slot $t$, $\boldsymbol{q_r^{t-1}} = \{x_r^{t-1}, y_r^{t-1}, h_r^{t-1}\}$ expresses the position of RIS-UAV at the end of time slot $t-1$, $\boldsymbol{\tau_t} = \{\tau_1^t, \tau_2^t, ..., \tau_N^t\}$ is the remaining execution time vector of tasks on client nodes when time slot $t$ begins, $C_{ECN}^t$ represents the available computation capacity of the edge computing node when time slot $t$ begins.

*2) Action Space:* In time slot $t$, the agent of the CPN controller can choose an action from the action space.

$$\boldsymbol{a(t)} = \{\boldsymbol{\alpha_t}, \boldsymbol{\eta_t}, \boldsymbol{\Phi_t}, \boldsymbol{v_r^t}, \boldsymbol{P_t}\}, \tag{30}$$

where $\boldsymbol{\alpha_t}$ is the task offloading decision vector, $\boldsymbol{\eta_t}$ is the computation capacity allocation vector, $\boldsymbol{\Phi_t}$ is the phase shift vector of RIS, $\boldsymbol{v_r^t}$ is the movement vector of the RIS-UAV, and $\boldsymbol{P_t}$ is the transmit power of client node $i$.

*3) System Reward:* Since the goal of the system is to minimize the overall consumption according to Eq.(27), the reward is defined as the negative of the sum consumption in Eq.(26)

$$\mathcal{R}(t) = -O_t. \tag{31}$$

### B. Proposed Adaptive N-Step Method Based On SAC Algorithm

This section introduces a decision-making method. Our method is developed based on the SAC algorithm, and we introduce an adaptive N-step learning trick to enhance the algorithm's efficiency.

*1) DRL Preliminaries:* The DRL is a combination of deep learning and reinforcement learning. The goal of reinforcement learning is to train an agent to effectively adapt to the environment and make decisions that maximize the reward, which can be formulated as an MDP.

In an MDP, the Q function is often used to evaluate the value of an action in a state, i.e., the expectation of return when an action is taken in a given state.

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_\pi\left[G_t \mid s_t = s, a_t = a\right] \\ &= \mathbb{E}_\pi\left[r_{t+1} + \gamma Q_\pi\left(s_{t+1}, a_{t+1}\right) \mid s_t = s, a_t = a\right], \end{aligned} \tag{32}$$

where $\gamma$ is the discount factor, $\pi$ is the current policy, $G_t$ is the return, which is defined as

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{33}$$

In addition, the value function is used to assess the value of a state and can be derived from the Q function and the policy $\pi$.

$$\begin{aligned} V_\pi(s) &= \sum_{a \in A} \pi(a \mid s) Q_\pi(s,a) \\ &= \mathbb{E}_\pi\left[r_{t+1} + \gamma V_\pi\left(s_{t+1}\right) \mid s_t = s\right]. \end{aligned} \tag{34}$$

In the process of reinforcement learning, an agent outputs an action based on the current state of the environment and its policy, the action is then executed and has an effect on the environment. Then the environment changes its state based on the action and returns a reward. Then the agent uses the received reward and new state to update its policy or value function to make better decisions in the future.

Typically, agents make decisions based on learned value functions or policies. Traditional reinforcement learning algorithms usually use tables to store state value functions or action value functions, but table methods are difficult to handle continuous state spaces and Markov problems with complex state spaces. To solve this problem, DRL combines deep learning with reinforcement learning. For example, the actor-critic algorithm uses two networks, the actor network for learning the policy function and the critic network for learning the value function.

*2) Soft Actor-Critic Architecture:* In this section, a brief introduction to the SAC algorithm is provided. The SAC algorithm is designed to learn a stochastic policy for continuous control tasks [32]. The selection of SAC was based on its stability through entropy maximization, which ensures robust exploration and avoids premature convergence to suboptimal policies. Furthermore, its off-policy efficiency allows for effective data reuse, making it highly sample-efficient. Additionally, SAC's design is particularly well-suited for continuous action spaces, making it an ideal choice compared to other algorithms.

To enhance the exploration efficiency of the SAC algorithm, the maximum entropy framework is leveraged. This framework introduces an entropy term into the reward function, which incentivizes the agent to explore more thoroughly, thereby reducing the convergence time of the algorithm.

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{\boldsymbol{s}_t, \boldsymbol{a}_t \sim \rho_\pi}\left[r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha \mathcal{H}(\pi(\cdot|\boldsymbol{s}_t))\right], \tag{35}$$

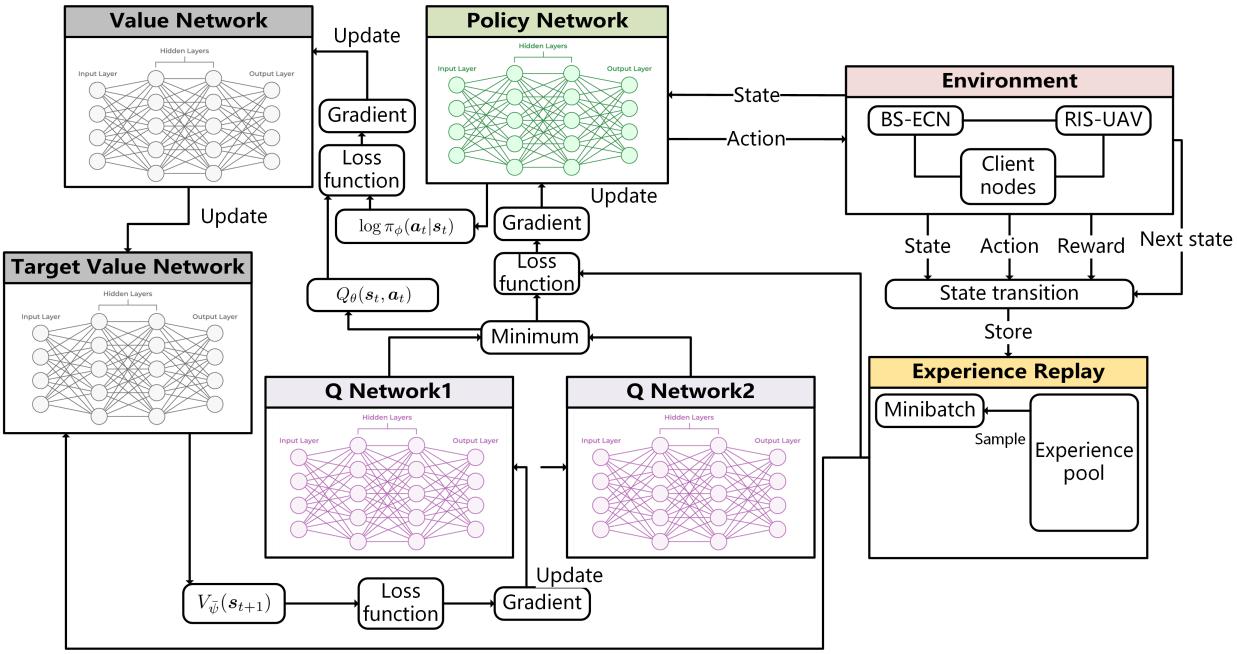


Fig. 3. Network architecture of SAC.

where $\alpha$ is the temperature parameter and $\mathcal{H}(\pi(\cdot|\boldsymbol{s}_t))$ is the entropy of the policy, which is defined as

$$\mathcal{H}(\pi(\cdot|\boldsymbol{s}_t)) = -\mathbb{E}_{\boldsymbol{a}_t\sim\pi(\cdot|\boldsymbol{s}_t)}\left[\log \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)\right]. \tag{36}$$

Then the Q value of the pair $(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is evaluated by the Bellman equation

$$\mathcal{T}^\pi Q(\boldsymbol{s}_t, \boldsymbol{a}_t) \triangleq r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[V(s_{t+1})], \tag{37}$$

The policy is updated according to the KL-divergence, which can be expressed as

$$\pi_{new} = \arg\min_{\pi'\in\Pi} D_{KL}\left(\pi'(\cdot|\boldsymbol{s}_t)||\frac{\exp(Q^{\pi_{old}}(\boldsymbol{s}_t,\cdot))}{Z^{\pi_{old}}(\boldsymbol{s}_t)}\right). \tag{38}$$

The parameters of soft value network is updated by minimizing $J_V(\psi)$

$$J_V(\psi) = \mathbb{E}_{\boldsymbol{s}_t\sim\mathcal{D}}\left[\frac{1}{2}\left(V_\psi(\boldsymbol{s}_t) - \mathbb{E}_{\boldsymbol{a}_t\sim\pi_\phi}\left[Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) - \log \pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)\right]\right)^2\right]. \tag{39}$$

In line with the above, gradient of Eq.(39) can be calculated using an unbiased estimator

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(\boldsymbol{s}_t)(V_\psi(\boldsymbol{s}_t) - Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) + \log \pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)). \tag{40}$$

The parameters of Q-function network can be trained by minimizing the Bellman residual

$$J_Q(\theta) = \mathbb{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) - \hat{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right)^2\right], \tag{41}$$

then we could get gradient of the Bellman residual in Eq.(41)

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)(Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) - r(\boldsymbol{s}_t, \boldsymbol{a}_t) - \gamma V_{\bar{\psi}}(\boldsymbol{s}_{t+1})). \tag{42}$$

The parameters of the policy network are updated by minimizing the KL-divergence in Eq.(38)

$$J_\pi(\phi) = \mathbb{E}_{\boldsymbol{s}_t\sim\mathcal{D}}\left[D_{KL}\left(\pi_\phi(\cdot|\boldsymbol{s}_t)\middle\|\frac{\exp(Q^{\pi_{old}}(\boldsymbol{s}_t,\cdot))}{Z^{\pi_{old}}(\boldsymbol{s}_t)}\right)\right]. \tag{43}$$

To minimize $J_\pi(\phi)$ in Eq.(43), an efficient way is to use the reparameterization trick, which is an unbiased gradient estimator that extends the policy gradient method to any tractable stochastic policy [32]

$$\boldsymbol{a}_t = f_\phi(\epsilon_t, \boldsymbol{s}_t), \tag{44}$$

where $\epsilon_t$ is a random variable with a fixed distribution, and $f_\phi$ is a deterministic function with parameters $\phi$. Then we could rewrite the gradient of the KL-divergence in Eq.(43) based on the reparameterization trick

$$\hat{\nabla}_\phi J_\pi(\phi) = \mathbb{E}_{\boldsymbol{s}_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}\left[\nabla_\phi \log \pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)\nabla_{\boldsymbol{a}_t} Q^{\pi_{old}}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]. \tag{45}$$

The network architecture of SAC is illustrated in Fig. 3.

*3) Soft Actor-Critic with Adaptive N-Step Learning:* The following section will introduce a revised form of the SAC, which utilizes the N-step return to get a more accurate estimate of the Q-value, thus improves the long-term performance of the algorithm [33].

In adaptive N-step learning, the Bellman operator $\mathcal{T}^\pi$ in Eq.(37) could be replaced by $\mathcal{T}^{\pi,N}$, which is defined as

$$\mathcal{T}^{\pi,N} Q(\boldsymbol{s}_t, \boldsymbol{a}_t) \triangleq \mathbb{E}[\sum_{n=1}^{N-1} \gamma^n r(\boldsymbol{s_{t+n}}, \boldsymbol{a_{t+n}}) + \gamma^N V(\boldsymbol{s_{t+N}})] + r(\boldsymbol{s}_t, \boldsymbol{a}_t), \tag{46}$$

where $N$ is the number of steps to look ahead.

The networks' parameters are updated after obtaining all of their gradients

$$\psi \leftarrow \psi - \alpha_\psi \hat{\nabla}_\psi J_V(\psi), \tag{47}$$

$$\theta_i \leftarrow \theta_i - \alpha_\theta \hat{\nabla}_{\theta_i} J_Q(\theta_i) for\ i \in \{1,2\}, \tag{48}$$

$$\phi \leftarrow \phi - \alpha_\phi \hat{\nabla}_\phi J_\pi(\phi), \tag{49}$$

$$\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi} \tag{50}$$

where $\alpha_\psi$, $\alpha_\theta$, $\alpha_\phi$ are the learning rates.

**Algorithm 1:** Adaptive N-Step Soft Actor-Critic (ANSSAC)

**Input:** initial $\phi$, $\theta_1$, $\theta_2$, $\psi$, environment
**Output:** optimal policy $\pi_\phi$

initialize target value network parameters $\bar{\psi} \leftarrow \psi$;
initialize replay buffer $\mathcal{D}$;
**for** *each iteration* **do**
  **for** *each environment step* **do**
    select action $\boldsymbol{a}_t \sim \pi_\phi(\cdot|\boldsymbol{s}_t)$ and observe reward $r_t$ and new state $\boldsymbol{s}_{t+1}$;
    store transition tuple $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ in replay buffer $\mathcal{D}$;
    update state $\boldsymbol{s}_t = \boldsymbol{s}_{t+1}$;
  **end**
  **for** *each gradient step* **do**
    sample batch $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ from $\mathcal{D}$;
    compute gradients using Eq.(40), Eq.(42), Eq.(45);
    update Q-function parameters using Eq.(48);
    update value network parameters using Eq.(47);
    update policy network parameters using Eq.(49);
    update target value network parameters using Eq.(50);
  **end**
**end**

*4) Complexity Analysis:* Assuming the time consumed by running steps 1 and 2 is $\xi_0$ and $\xi_1$, respectively. These two steps only run one time. For step 3, the outer loop runs $K$ times with an average time consumption of $\xi_2$.

In each iteration of the outer loop of step 3, there are two nested inner loops. The first inner loop for environment steps runs $M$ times. The average time consumption for one environment step is $\xi_{2,1}$,

$$\xi_{2,1} = \xi_{2,1,1} + \xi_{2,1,2} + \xi_{2,1,3}, \tag{51}$$

where $\xi_{2,1,1}$, $\xi_{2,1,2}$, $\xi_{2,1,3}$ are the time consumptions for selecting action and observing, storing transition tuple, and updating state respectively in each environment step.

The second inner loop for gradient steps runs $N$ times. The average time consumption for one gradient step is $\xi_{2,2}$,

$$\xi_{2,2} = \xi_{2,2,1} + \xi_{2,2,2} + \xi_{2,2,3} + \xi_{2,2,4} + \xi_{2,2,5} + \xi_{2,2,6}, \tag{52}$$

where $\xi_{2,2,1}$, $\xi_{2,2,2}$, $\xi_{2,2,3}$, $\xi_{2,2,4}$, $\xi_{2,2,5}$, $\xi_{2,2,6}$ are the time consumptions for sampling from $\mathcal{D}$, computing gradients, updating Q-function parameters, updating value network parameters, updating policy network parameters, and updating target value network parameters respectively in each gradient step.

The total time complexity of the algorithm is given by:

$$T_{Algorithm} = \xi_0 + \xi_1 + K \times (M \times \xi_{2,1} + N \times \xi_{2,2}). \tag{53}$$

In big-O notation, ignoring the constant time consumptions $\xi_0$ and $\xi_1$ and assuming $\xi_{2,1}$ and $\xi_{2,2}$ are constants, the time complexity can be expressed as $O(K \times (M+N))$. This indicates that the running time of the algorithm is proportional to the product of the number of outer loop iterations $K$ and the sum of the number of environment steps $M$ and the number of gradient steps $N$.

## V. Simulation Results

In this section, we evaluate the performance of the proposed ANSSAC method.

### A. Simulation Setup

We conducted the simulations in Python 3.7.15, TensorFlow 1.14.0, Gym 0.21.0, NumPy 1.19.5. To prove the effectiveness of the ANSSAC method, we compare the following decision making baselines with the proposed ANSSAC method:

- Local Computing (LC): When a client node receives a task, it will execute the task locally using its own computation resources.
- Full Offloading (FO): In contrast with LC, all tasks of client nodes are completely transferred to the ECN for processing.
- Random Offloading (RO): When a client node receives a task, it is randomly selected to run locally or to be offloaded to the ECN.
- Asynchronous Advantage Actor Critic (A3C): In this method, a decision-making agent is trained using the A3C algorithm in [34] and deployed. When a time slot starts, the agent makes the decision based on the current state of the system.
- Soft Actor-Critic (SAC): A decision-making agent is trained using SAC algorithm proposed in [32]. The decision-making agent is deployed in the system to make decisions at the beginning of each time slot.

The BS antenna is positioned at coordinates (0, 0, 10) with a gain of 20 dBi [27]. The ECN has a computational capacity of 5000 million CPU cycles per second, and an energy efficiency parameter of $10^{-26}$ [27]. The UAV starts at (0, 0, 0) and can move a maximum distance of 10 per time slot, with a weight of 2 kg. The computational consumption, transmission consumption, and time constraints of tasks follow a Gaussian distribution. In order to meet the requirements of compute-intensive tasks considered in this paper, tasks have an average transmission size of 450 KB with a variance of 100 KB, an average computing consumption of 450 million CPU cycles with a variance of 100 million CPU cycles [30], and an average task time limit of 10 seconds with a variance of 2 seconds. Tasks arrive at a rate of 0.3 tasks per time slot. The simulation includes 5 client nodes, each with a computational capacity of 1000 million CPU cycles per second by default, an idle power of 0.1 watts, and a maximum transmit power of 5 watts [30]. The energy efficiency parameter for client nodes is $10^{-26}$, and their antenna gain is 20 dBi. The time interval

TABLE II: ALGORITHM PARAMETERS

| Parameters | Value |
|---|---|
| Learning rate of actor | 0.003 |
| Learning rate of critic | 0.003 |
| Entropy temperature | 0.0003 |
| Discount factor | 0.99 |
| Replay buffer size | 10000 |
| Batch size | 256 |
| Environment steps per episode | 1000 |
| Gradient steps per episode | 1000 |

between time slots is 5 seconds. The noise power density is -174 dBm/Hz [27], the carrier frequency is $1 * 10^{13}$ Hz with a carrier wavelength of $3 * 10^{-5}$ meters, and the transmission bandwidth is $4 * 10^{11}$ Hz [21]. The preference of latency $\iota$ is 0.5 [30].

In the ANSSAC method, the learning rate of the networks is $3*10^{-3}$, the entropy temperature is $3*10^{-4}$, the discount factor is 0.99, the replay buffer size is 10000, the batch size is 256, the environment steps per episode is 1000, and the gradient steps per episode is 1000. Since the time taken for decision making and inference is relatively minimal, we have ignored the impact of the algorithm decision making time. Tables II and III summarized the algorithm and simulation parameters, respectively.

### *B. Convergence Analysis*

In this section, the convergence of the proposed method is analysed. In Fig. 4, we present a comparison of the proposed method's convergence at various learning rates. The convergence of the algorithm is slower at a learning rate of $3 * 10^{-4}$, this may be due to inefficient learning as a result of low learning rates. When the learning rate is $3 * 10^{-2}$, the training results produce oscillations, which may be due to the fact that the algorithm fails to converge because the learning rate is too large. Thus, it can be concluded that the learning rate of $3 * 10^{-3}$ is the most suitable for the proposed method.

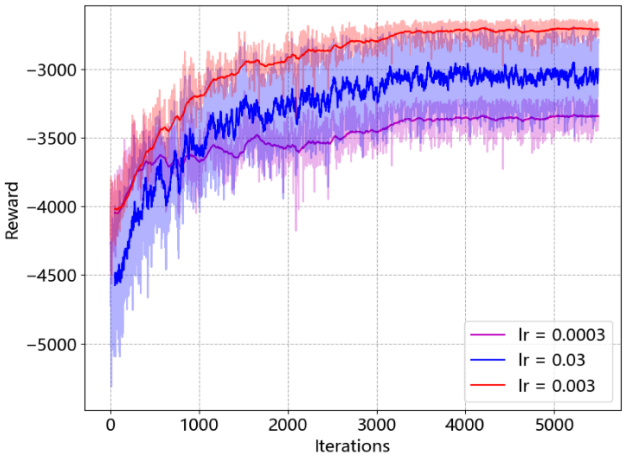


Fig. 4. Convergence comparison under different learning rates.

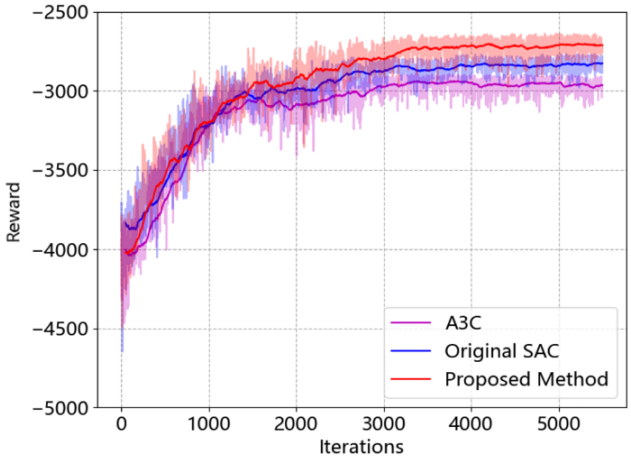


Fig. 5. Convergence comparison.

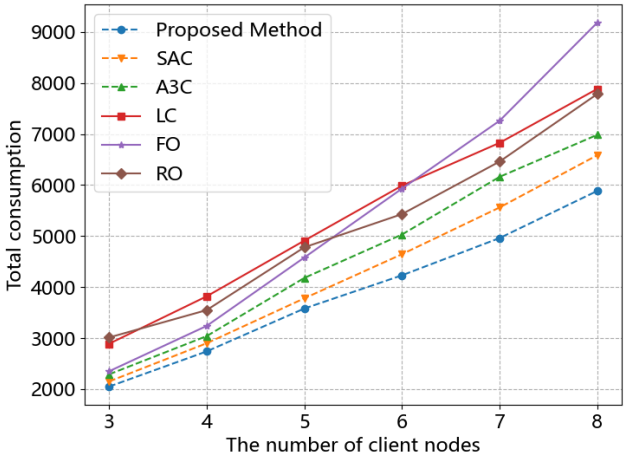


Fig. 6. Total consumption comparison under different numbers of client nodes.

In Fig. 5, the convergence of ANSSAC is compared with original SAC algorithm and A3C algorithm. From the graph, it can be concluded that the reward of all algorithms increases as the training process continues. The need to calculate the n-step rewards in advance can lead to larger errors at the beginning of training, thus the performance of the proposed method is not as good as the existing methods in the initial stage. The proposed method converges at about 1000 episodes, while the original SAC and A3C algorithm converge at about 2000 episodes. The convergence value of the proposed method is approximately -2700, whereas the convergence values for the original SAC and A3C algorithms are around -2810 and -2920, respectively. This indicates that ANSSAC achieves a higher convergence value compared to SAC and A3C Thus, it can be concluded that ANSSAC converges faster and achieves a higher convergence value than the original SAC and A3C algorithms.

TABLE III: SIMULATION PARAMETERS

| Parameters | Value |
|---|---|
| Position of BS antenna | (0, 0, 10) |
| BS antenna gain | 20 dBi |
| Computation capacity of ECN | 5000M CPU cycles/s |
| Energy efficiency parameter of ECN | $10^{-26}$ |
| UAV start coordination | (0, 0, 0) |
| UAV max movement distance within a time slot | 10 |
| Weight of RIS-UAV | 2 kg |
| Average of task transmission size | 450KB |
| Variance of task transmission size | 100KB |
| Average of task computing consumption | 450M CPU cycles |
| Variance of task computing consumption | 100M CPU cycles |
| Average of task time limit | 10 s |
| Variance of task time limit | 2 s |
| Task arrival rate | 0.3 |
| Number of client nodes | 5 |
| Computation capacity of client nodes | 1000M CPU cycles/s |
| Idle power of client nodes | 0.1 W |
| Maximum transmit power of client nodes | 5 W |
| Energy efficiency parameter of client nodes | $10^{-26}$ |
| Client node antenna gain | 20 dBi |
| Time interval between time slots | 5 |
| Noise power density | - 174 dBm/Hz |
| Carrier frequency | $1 * 10^{13}$ |
| Carrier wavelength | $3 * 10^{-5}$ |
| Transmission bandwidth | $4 * 10^{11}$ |

### C. Performance Evaluation

In this section, the performance of the proposed ANSSAC method is evaluated in various environments.

The effectiveness of the proposed ANSSAC method is compared with above mentioned baselines for different numbers of client nodes in Fig. 6. The ECN's computational capacity is 5000 million CPU cycles per second, and the client nodes' computational capacity is set to 1000 million CPU cycles per second. The vertical axis represents the overall consumption over 1000 time slots. It can be concluded that the increase of client node leads to the increase of the task count, which leads to the increasement of system consumption. Furthermore, the proposed ANSSAC method exhibits the lowest total consumption among all the methods, thereby demonstrating its efficiency.

Fig. 7 shows the comparison of consumption for the ANSSAC method and other baselines across different computation capacities of client nodes. From the figure, it can be observed that an increase in ECN computational power results in a decrease in the time and energy consumed by tasks offloaded to the ECN, thereby improving the performance of all methods except the LC method. The FO method is particularly impacted because low computational power in the ECN leads to a high number of task timeouts, incurring significant timeout penalties. As the computational power of the ECN increases, the timeout rate for the FO method decreases substantially. Although the FO method does not perform as well as the LC and RO methods when the computational power is below 3,000 MHz, it outperforms them once the computational power exceeds 4,000 MHz. Furthermore, the performance of the FO method approaches that of the A3C algorithm when the ECN computational power reaches 7,000 MHz. The figure demonstrates that the ANSSAC method outperforms other

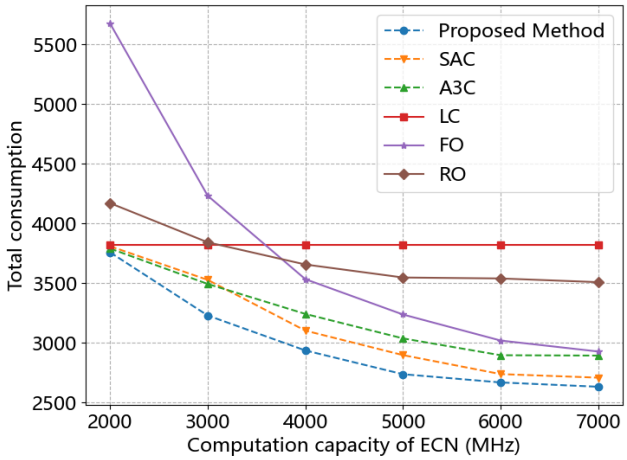


Fig. 7. Total consumption comparison under different computation capacity of edge computing node.

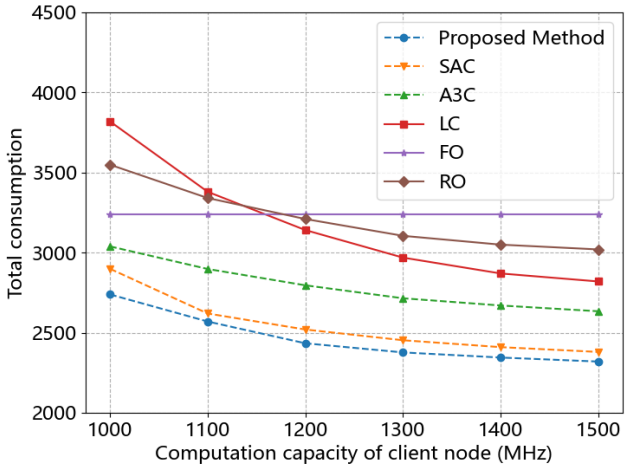


Fig. 8. Total consumption comparison under different computational capacity of client nodes.

baselines, thereby confirming its superior performance.

In Fig. 8, the ANSSAC method is compared with the other baselines in different computation capacities of client nodes. From the figure, it is evident that as local computing power increases, the consumption for processing tasks locally decreases. The LC method, which relies entirely on local devices for computation, is the most affected by this increase. Consequently, the total consumption of the LC method decreases the fastest as local computing power rises. In contrast, the FO method, which does not utilize local devices for computation, remains unaffected by changes in local computing power. Notably, the total consumption of the LC method becomes less than that of the FO method when local computation power exceeds 1200 MHz. Compared to other methods, the proposed ANSSAC method is the most efficient method under different computing power of client nodes.

In Fig. 9, the performance of the ANSSAC method and other baselines with various quantities of reflecting elements

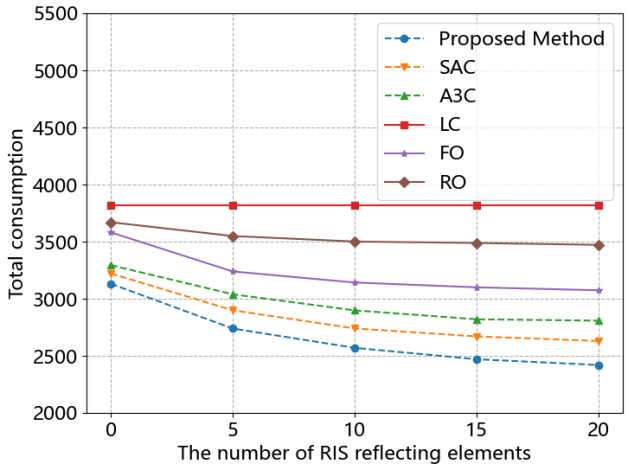


Fig. 9. Total consumption comparison under different reflecting unit number.

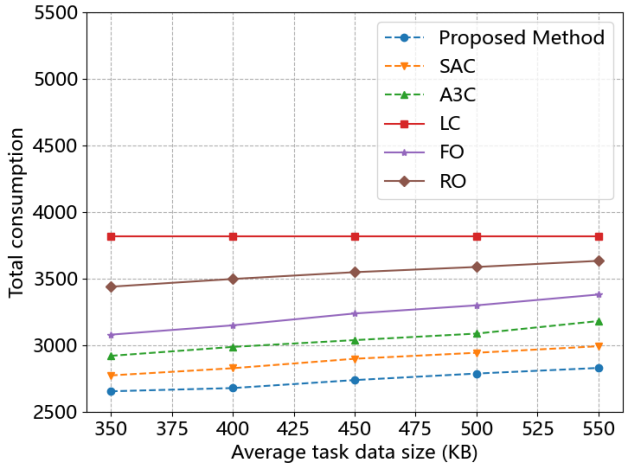


Fig. 10. Total consumption comparison under different task size.

is compared. From the figure, it is evident that the FO method, which offloads all tasks to the ECN for computation, benefits significantly from an increase in the number of reflecting elements. This increase substantially improves communication efficiency due to the high volume of data transmission involved. In contrast, the LC method exhibits no change in consumption with the increase in reflecting elements, as it does not rely on data transmission.

Fig. 10 shows the performance comparison with different average sizes of the tasks. With the increase of average task size, the transmission time and energy consumption would increase, thus increase the total consumption under methods that needed to offload tasks to the ECN. The proposed method exhibits a lower consumption than other baselines, its efficiency is due to the dynamic optimization of task allocation and computational resource utilization in complex environments.

In Fig. 11, the system consumption using the ANSSAC method and other baselines is compared under different computational complexities. It is observed that an increase in com-

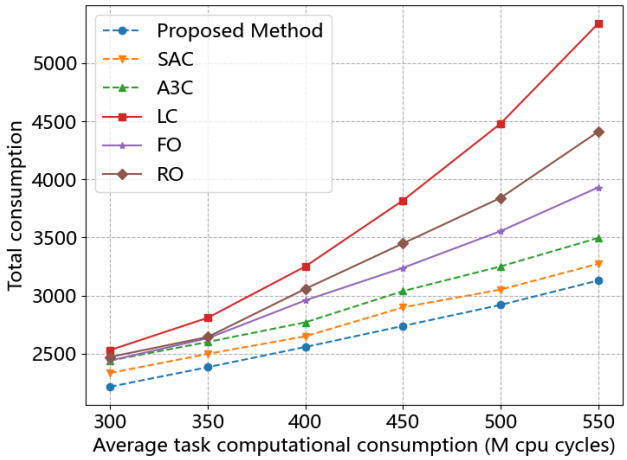


Fig. 11. Total consumption comparison under different task complexity.

putational complexity results in higher execution consumption of tasks across all methods. The LC method exhibits the highest total consumption, which grows exponentially with the increase in computational complexity. This method struggles to process incoming tasks promptly using only local devices, leads to longer waiting times for subsequent tasks and a higher rate of task timeouts. Similarly, both FO and RO demonstrate a trend toward higher total consumption with increasing computational complexity, although the exact nature of this trend is not specified in the figure. In contrast, the DRL methods show relatively slow growth in total consumption as computational complexity rises. This is attributed to the efficient allocation of task offloading, which enables a more rational distribution of tasks and, consequently, more manageable consumption growth. The ANSSAC method exhibits lower consumption than other baselines, thereby demonstrating its superiority.

## VI. Conclusions and Future Works

In this paper, the CPN architecture based on RIS-UAV-aided NOMA-THz communication was proposed to meet the increasing computational demands for processing large volumes of real-time data for IoT applications. The CPN was intended to assist IoT devices in task processing by enabling task offloading to ECNs and rationally allocating computational resources. To enhance communication efficiency, THz communication combined with the NOMA technique was utilized. To address the challenges of obstacle occlusion and attenuation in THz communication, the RIS-UAV was employed to support the communication process. To allocate computational and communication resources more efficiently in the proposed system, an optimization problem was introduced, aimed at minimizing the system's total consumption, encompassing both time and energy consumption. Considering the features of the system, the optimization problem was formulated as an MDP, and an adaptive N-Step method based on SAC algorithm was adopted to minimize the overall consumption. The simulation results showed that the proposed method was an effective solution to reduce the energy and time consumption of the

proposed CPN architecture. Future researches will concentrate on several essential areas, such as the incorporation of cloud computing and consideration of blockages.

## REFERENCES


[1] P. Upadhyaya, S. Dutt, Ruchi, and S. Upadhyaya, "6G Communication: Next Generation Technology for IoT Applications," in *2021 First International Conference on Advances in Computing and Future Communication Technologies (ICACFCT)*. Meerut, India, Dec. 2021, pp. 23–26.

[2] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, O. Dobre, and H. V. Poor, "6G Internet of Things: A Comprehensive Survey," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 359–383, Jan. 2022.

[3] N. H. Chu, D. T. Hoang, D. N. Nguyen, N. Van Huynh, and E. Dutkiewicz, "Joint Speed Control and Energy Replenishment Optimization for UAV-Assisted IoT Data Collection with Deep Reinforcement Transfer Learning," *IEEE Internet Things J.*, vol. 10, no. 7, pp. 5778–5793, Apr. 2023.

[4] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-Preserving Blockchain-Based Federated Learning for IoT Devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Feb. 2021.

[5] M.-D. Cano, A. Guillen-Perez, I. Tasic, and A. Villafranca, "A Conceptual Framework for the Development of Autonomous Driving in 6G: The Role of AI and Edge Computing," in *2023 8th International Conference on Control, Robotics and Cybernetics (CRC)*, Dec. 2023, pp. 97–105.

[6] N. H. Mahmood, G. Berardinelli, E. J. Khatib, R. Hashemi, C. De Lima, and M. Latva-aho, "A Functional Architecture for 6G Special-Purpose Industrial IoT Networks," *IEEE Trans. Ind. Informat.*, vol. 19, no. 3, pp. 2530–2540, Mar. 2023.

[7] X. Gong, C. Bai, S. Ren, J. Wang, and C. Wang, "A Survey of Compute First Networking," in *2023 IEEE 23rd International Conference on Communication Technology (ICCT)*. Wuxi, China, Oct. 2023, pp. 688–695.

[8] C. Unicom, "Compute First Network White Paper," White Paper, China Unicom, Nov. 2019.

[9] M. Inomata, W. Yamada, N. Kuno, M. Sasaki, K. Kitao, M. Nakamura, H. Ishikawa, and Y. Oda, "Terahertz Propagation Characteristics for 6G Mobile Communication Systems," in *2021 15th European Conference on Antennas and Propagation (EuCAP)*. Dusseldorf, Germany, Mar. 2021, pp. 1–5.

[10] Y. Zhang, H.-M. Wang, Q. Yang, and Z. Ding, "Secrecy Sum Rate Maximization in Non-Orthogonal Multiple Access," *IEEE Commun. Lett.*, vol. 20, no. 5, pp. 930–933, May 2016.

[11] J. Li, L. Yang, Q. Wu, X. Lei, F. Zhou, F. Shu, X. Mu, Y. Liu, and P. Fan, "Active RIS-Aided NOMA-Enabled Space-Air-Ground Integrated Networks With Cognitive Radio," *IEEE Journal on Selected Areas in Communications*, vol. 43, no. 1, pp. 314–333, 2025.

[12] Y. Zhang, L. Yang, X. Li, K. Guo, and H. Liu, "Covert Communications for STAR-RIS-Assisted Industrial Networks With a Full Duplex Receiver and RSMA," *IEEE Internet of Things Journal*, vol. 11, no. 12, pp. 22 483–22 493, 2024.

[13] Q. Wu and R. Zhang, "Intelligent Reflecting Surface Enhanced Wireless Network via Joint Active and Passive Beamforming," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5394–5409, Nov. 2019.

[14] Z. Yang, Y. Li, H. Liu, and F. He, "Federated Reinforcement Learning for RIS-Aided Non-Orthogonal Multiple Access MEC," in *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*. London, United Kingdom, Sep. 2022, pp. 1–5.

[15] T.-H. Nguyen, H. Park, and L. Park, "Recent Studies on Deep Reinforcement Learning in RIS-UAV Communication Networks," in *2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. Bali, Indonesia, Feb. 2023, pp. 378–381.

[16] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Commun. Surv. Tutor.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart. 2017.

[17] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing Systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*. Barcelona, Spain, Jul. 2016, pp. 1451–1455.

[18] T. P. Truong, V. D. Tuong, N.-N. Dao, and S. Cho, "FlyReflect: Joint Flying IRS Trajectory and Phase Shift Design Using Deep Reinforcement Learning," *IEEE Internet Things J.*, vol. 10, no. 5, pp. 4605–4620, Mar. 2023.

[19] X. Liu, Y. Yu, B. Peng, X. B. Zhai, Q. Zhu, and V. C. M. Leung, "RIS-UAV Enabled Worst-Case Downlink Secrecy Rate Maximization for Mobile Vehicles," *IEEE Trans. Veh. Technol.*, vol. 72, no. 5, pp. 6129–6141, May 2023.

[20] H. Zhang, M. Huang, H. Zhou, X. Wang, N. Wang, and K. Long, "Capacity Maximization in RIS-UAV Networks: A DDQN-Based Trajectory and Phase Shift Optimization Approach," *IEEE Trans. Wireless Commun.*, vol. 22, no. 4, pp. 2583–2591, Apr. 2023.

[21] I. F. Akyildiz, C. Han, Z. Hu, S. Nie, and J. M. Jornet, "Terahertz Band Communication: An Old Problem Revisited and Research Directions for the Next Decade," *IEEE Trans. Commun.*, vol. 70, no. 6, pp. 4250–4285, Jun. 2022.

[22] Z. Li, Z. Chen, X. Ma, and W. Chen, "Channel Estimation for Intelligent Reflecting Surface Enabled Terahertz MIMO Systems: A Deep Learning Perspective," in *2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*. Chongqing, China, Aug. 2020, pp. 75–79.

[23] W. Xun, Y. Liu, Y. Ni, H. Zhao, Z. Xu, and X. Ge, "A Survey on Terahertz Communication Theory Assisted by Intelligent Reflecting Surface and Device-to-Device Technologies," in *2022 IEEE 22nd International Conference on Communication Technology (ICCT)*. Nanjing, China, Nov. 2022, pp. 678–682.

[24] H. Wang, H. Zhang, X. Liu, K. Long, and A. Nallanathan, "Joint UAV Placement Optimization, Resource Allocation, and Computation Offloading for THz Band: A DRL Approach," *IEEE Wirel. Commun. Lett.*, vol. 22, no. 7, pp. 4890–4900, Jul. 2023.

[25] S. Rezwan, S. Shin, and W. Choi, "Efficient User Clustering and Reinforcement Learning Based Power Allocation for NOMA Systems," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. Jeju, Korea (South), Oct. 2020, pp. 143–147.

[26] L. Qian, Y. Wu, J. Ouyang, Z. Shi, B. Lin, and W. Jia, "Latency Optimization for Cellular Assisted Mobile Edge Computing via Non-Orthogonal Multiple Access," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5494–5507, May 2020.

[27] L. Guo, J. Jia, J. Chen, A. Du, and X. Wang, "Joint Task Offloading and Resource Allocation in STAR-RIS Assisted NOMA Ssystem," in *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*. London, United Kingdom, Sep. 2022, pp. 1–5.

[28] Y. Pan, K. Wang, C. Pan, H. Zhu, and J. Wang, "UAV-Assisted and Intelligent Reflecting Surfaces-Supported Terahertz Communications," *IEEE Wireless Commun. Lett.*, vol. 10, no. 6, pp. 1256–1260, Mar. 2021.

[29] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation Offloading in Multi-access Edge Computing: A Multi-Task Learning Approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2745–2762, May 2019.

[30] W. Li, X. Chen, L. Jiao, and Y. Wang, "Deep Reinforcement Learning-Based Intelligent Task Offloading and Dynamic Resource Allocation in 6G Smart City," in *2023 IEEE Symposium on Computers and Communications (ISCC)*. Gammarth, Tunisia, Jul. 2023, pp. 575–581.

[31] H. Guo and J. Liu, "UAV-Enhanced Intelligent Offloading for Internet of Things at the Edge," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2737–2746, Nov. 2019.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *International conference on machine learning (ICML)*. PMLR, Jan. 2018, pp. 1861–1870.

[33] X. Qiu, W. Zhang, W. Chen, and Z. Zheng, "Distributed and Collective Deep Reinforcement Learning for Computation Offloading: A Practical Perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1085–1101, May 2021.

[34] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," in *International Conference on Machine Learning (ICML)*. PMLR, Feb. 2016, pp. 1928–1937.