Computation Offloading Optimization in Satellite–Terrestrial Integrated Networks via Offline Deep Reinforcement Learning

Bo Xie, Haixia Cui, *Senior Member, IEEE*, Peng Cao, Yejun He, *Senior Member, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*—As the demand for global Internet connectivity continues to grow, the satellite–terrestrial integrated networks (STINs) have become more and more crucial for expanding the service coverage and enhancing the network performance. However, the task offloading problem in STINs faces many significant challenges, such as high processing latency and energy consumption. The current intelligent offloading strategies often rely on the real-time interactions with the environments which not only consume valuable satellite resources but also cause irreversible damage to the satellite equipment due to some operational errors. To address these issues, in this article, we propose an offline deep reinforcement learning (offline DRL) approach to learn and optimize the task offloading decisions by leveraging the stored historical decision data and employing the soft actor–critic (SAC) algorithm specifically. Experimental results show that the proposed strategy outperforms most of the existing methods in terms of latency and energy consumption and effectively reduces the direct interactions with STINs.**

***Index Terms*—Offline deep reinforcement learning (offline DRL), satellite–terrestrial integrated networks (STINs), soft actor–critic (SAC), task offloading.**

I. Introduction

WITH the rapid development of aerospace technology, the satellite–terrestrial integrated networks (STINs) have shown key roles in providing for the global connectivity. It not only expands the traditional terrestrial network coverage but also optimizes the data processing through edge computing which can reduce the data processing delays and enhance the real-time applications [1]. Moreover, in the remote or emergency areas, the local computing resource scarcity often limits some task executions which then require more satellite support [26]. At this point, the satellite edge task offloading transferring tasks from ground users to near Earth satellites becomes crucial for improving the network performance and user experience which can effectively mitigate the geographical constraints on information service quality and enhance the user experience in remote areas [2].

However, the existing satellite edge task offloading technique still encounters some challenges, such as processing latency and energy consumption [3]. To address these issues, the researchers have employed many traditional optimization algorithms [4], [5], [6] and deep reinforcement learning (DRL) techniques [7], [8], [29]. The traditional algorithms solve the problems with precise mathematical models and offer strong theoretical supports [6]. But, despite the reliability, the traditional algorithms cannot perform well in the dynamic and unpredictable environments due to their dependence on the predefined models that assume static conditions [18], [19]. DRL, on the other hand, adaptively learns and formulates the optimal decisions through the real-time environmental interactions, excelling in the dynamic environments [8]. Its ability to learn directly from the ongoing interactions with the environments without pre-established models makes it especially suited to the environments characterized by uncertainty and changing [18].

Despite the superiority of DRL in artificial intelligence, it still exists some limitations in STINs. In particular, it relies on the extensive real-time interactions with the environments to obtain the training data [31]. The process not only consumes valuable satellite resources but also causes irreversible damage to the satellite equipment due to the operational errors. For example, during the interaction learning process between DRL and STINs, the incorrectly assigned computational tasks to the satellite processor will lead to processor overload and overheating which may shorten its lifespan [27]. Fortunately, the digital twin technology can help to simulate the STINs and reduce the equipment using although DRL still needs some substantial real-time interactions for the optimal task offloading [9], [10].

From the above analysis, this article proposes a novel computation offloading optimization method based on the offline DRL (offline DRL) to minimize the direct interactions with the environments by utilizing the stored historical decision

Manuscript received 30 June 2024; revised 25 July 2024 and 17 August 2024; accepted 1 September 2024. Date of publication 6 September 2024; date of current version 20 November 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFE0107900; in part by the National Natural Science Foundation of China under Grant 61871433, Grant 61828103, Grant 61201255, and Grant 62071306; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515012052; and in part by the Research Platform of South China Normal University. *(Corresponding author: Haixia Cui.)*

Bo Xie, Haixia Cui, and Peng Cao are with the School of Electronics and Information Engineering, South China Normal University, Foshan 528225, China (e-mail: bo.xie@m.scnu.edu.cn; cuihaixia@m.scnu.edu.cn; caop@scnu.edu.cn).

Yejun He is with the College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: yjhe@szu.edu.cn).

Mohsen Guizani is with the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE (e-mail: mguizani@ieee.org).

Digital Object Identifier 10.1109/JIOT.2024.3455319

data and soft actor–critic (SAC) algorithm. Specifically, this proposed method can learn directly from the historical decision data and enhance the decision making through the policy iteration, thereby conserving the resources without compromising the performance of STINs during the DRL algorithm learning process. Initially, we use the table-based reinforcement learning (RL) [11] to collect the decision data and establish the historical data set. Subsequently, we apply the SAC algorithm to analyze and learn from this data. Finally, the trained SAC model is deployed in the same environment to verify its performance. From the perspective of multiagent RL [9], [12], our approach can be viewed as a special case of the "centralized learning, distributed execution" paradigm, where it learns from a historical data set rather than direct environmental interactions.

In brief, the main contributions of this article are summarized as follows.

1) We propose a satellite edge task offloading strategy based on offline DRL which significantly reduces the need for environmental interactions.
2) We effectively resolve the task offloading issues in STINs using the historical decision data and SAC algorithm.
3) Experimental validation demonstrates that the proposed strategy outperforms most of the existing methods in terms of task processing latency and energy consumption. The results not only advance the research of STINs but also offer new insights and tools for future management and optimization of STINs.

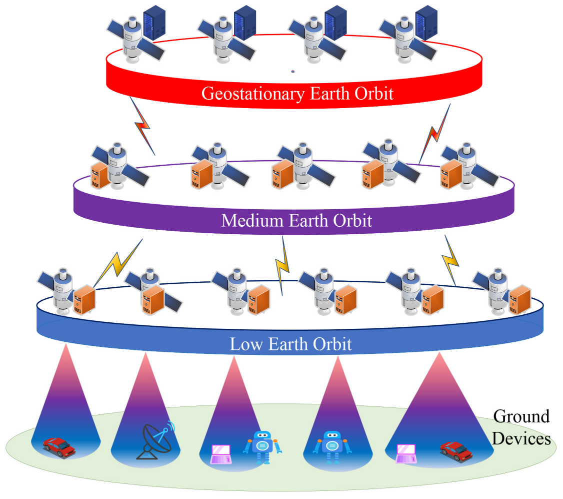
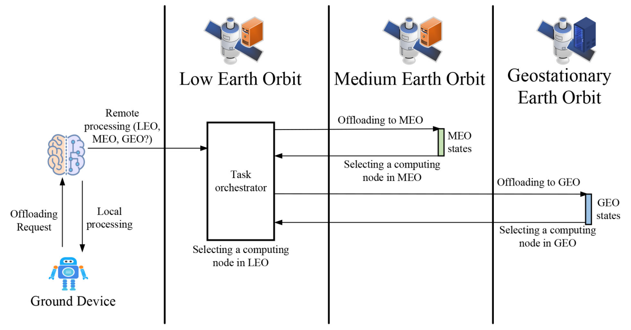


Fig. 1. STIN architecture.


Low Earth Orbit
Medium Earth Orbit
Geostationary Earth Orbit
Remote processing (LEO, MEO, GEO?)
Offloading Request
Local processing
Ground Device
Task orchestrator
Selecting a computing node in LEO
Offloading to MEO
MEO states
Selecting a computing node in MEO
Offloading to GEO
GEO states
Selecting a computing node in GEO
Fig. 2. Task offloading process.

II. System Model

As illustrated in Fig. 1, we consider a three-tier architecture consisting of low-Earth orbit (LEO), medium-Earth orbit (MEO), and geostationary orbit (GEO) satellites. The ground users communicate directly with the LEO satellites using dedicated antennas [20]. The direct communication between ground users and MEO or GEO satellites is infeasible due to the transmission distance, which necessitates the relays by LEO or MEO satellites [13], [14]. The satellites within the same orbital layer utilize laser technology for communications to ensure a very high transmission rate with negligible delay [21]. Conversely, the transfer of data between satellites in different orbital layers is managed using Ka-band technology [5]. The sets of LEO, MEO, and GEO satellites are defined as $\mathcal{M} = \{1, 2, \ldots, m, \ldots, M\}$, $\mathcal{N} = \{1, 2, \ldots, n, \ldots, N\}$, and $\mathcal{G} = \{1, 2, \ldots, g, \ldots, G\}$, respectively.

Consider a user $u_i = \{f_{i,\text{mips}}\}$ with a task to offload, denoted as $\tau = \{c_i, d_{i,\text{in}}, d_{i,\text{out}}\}$, where $f_{i,\text{mips}}$ is the MIPS of user u_i and c_i represents the MIPS required for task completion. $d_{i,\text{in}}$ and $d_{i,\text{out}}$ indicate the task input and output data sizes, respectively. The task offloading decision is represented by $\boldsymbol{x} = \{x_1, x_2, x_3, x_4\}$, where $\sum_{i=1}^{4} x_i = 1$ signifies that the task can only be executed locally ($x_1 = 1$) or offloaded to a specific orbital layer, i.e., $x_2 = 1$ for LEO, $x_3 = 1$ for MEO, $x_4 = 1$ for GEO. If the remote offloading is chosen by the ground user, the task scheduler on the LEO satellite selects the specific satellite in the LEO, MEO, or GEO layers to receive the tasks generated by the ground user. For example, as shown in Fig. 2, if a user decides to offload a task to the MEO layer, the LEO satellite's task scheduler will select an appropriate satellite from the MEO layer as the final offloading destination, after which the user awaits the return of the task results.

A. Communication Model

The communication process encompasses the data transfers from users to LEO satellites and the interorbital communications among satellites across three orbital layers. The transmission rate from users to LEO satellite is calculated by [5]

$$r_{i,m}^{U} = \log\left(1 + \frac{p_{i,m}^{U}|h_{i,m}^{U}|^2}{\delta_{i,m}^2}\right) b_{i,m}^{U} \tag{1}$$

where $p_{i,m}^{U}$, $|h_{i,m}^{U}|^2$, $\delta_{i,m}^2$, and $b_{i,m}^{U}$ denote the transmission power to LEO satellite, channel gain, noise power, and channel bandwidth, respectively.

For the communications from LEO to MEO and from MEO to GEO, a similar model can be employed to estimate their transmission rates and possible differences in transmission power, channel gain, and bandwidth [25]. Specifically, the communication rate from LEO to MEO can be given by

$$r_{m,n}^{LM} = \log\left(1 + \frac{p_{m,n}^{LM}|h_{m,n}^{LM}|^2}{\delta_{m,n}^2}\right) b_{m,n}^{LM}. \tag{2}$$

The communication rate from MEO to GEO can be written by

$$r_{n,g}^{MG} = \log\left(1 + \frac{p_{n,g}^{MG}|h_{n,g}^{MG}|^2}{\delta_{n,g}^2}\right) b_{n,g}^{MG} \tag{3}$$

where m, n, and g correspond to the indices of LEO, MEO, and GEO satellites, respectively.

B. Latency Model

If the task is executed locally, the system delay only includes the task execution part. If it is offloaded to a satellite, the delay includes both transmission and execution parts. Of course, the delay for local execution can be given by

$$D_i^{\text{loc}} = \frac{c_i}{f_{i,\text{mips}}^{\text{loc}}}. \tag{4}$$

When the offloading is taken by a satellite, the delay consists of both transmission and execution delays by depending on the specific orbital layer. For example, the system delay when offloading to the LEO satellite can be written by

$$D_i^{\text{LEO}} = \frac{c_i}{f_{\text{mips}}^{leo}} + \frac{d_{i,\text{in}}}{r_{i,m}^U} + \frac{d_{i,\text{out}}}{r_{m,i}^U} \tag{5}$$

where $r_{m,i}^U$ represents the downlink transmission rate from LEO to user.

The system delay when offloading to the MEO satellite is

$$D_i^{MEO} = \frac{c_{\text{mips}}}{f_{\text{mips}}^{\text{meo}}} + \frac{d_{i,\text{in}}}{r_{i,m}^U} + \frac{d_{i,\text{in}}}{r_{m,n}^{LM}} + \frac{d_{i,\text{out}}}{r_{n,m}^{LM}} + \frac{d_{i,\text{out}}}{r_{m,i}^U}. \tag{6}$$

Similarly, the system delay when offloading to the GEO satellite is

$$\begin{aligned} D_i^{\text{GEO}} = {} & \frac{c_{\text{mips}}}{f_{\text{mips}}^{\text{geo}}} + \frac{d_{i,\text{in}}}{r_{i,m}^U} + \frac{d_{i,\text{in}}}{r_{m,n}^{LM}} + \frac{d_{i,\text{in}}}{r_{n,g}^{MG}} \\ & + \frac{d_{i,\text{out}}}{r_{g,n}^{MG}} + \frac{d_{i,\text{out}}}{r_{n,m}^{LM}} + \frac{d_{i,\text{out}}}{r_{m,i}^U}. \end{aligned} \tag{7}$$

Then, the total task delay is calculated by

$$\begin{aligned} D_i = {} & x_1 \cdot D_i^{\text{loc}} + x_2 \cdot D_i^{\text{LEO}} + x_3 \cdot D_i^{MEO} + x_4 \cdot D_i^{\text{GEO}} \\ & \sum_{i=1}^{4} x_i = 1. \end{aligned} \tag{8}$$

Thus, the overall system delay can be obtained as

$$D = \sum_{i=1}^{|U|} D_i \tag{9}$$

where $|U|$ is the number of ground users. We assume that each user only offloads one task at time slot t.

C. Energy Consumption Model

The system energy consumption exists in the CPU part and the wireless transmission part. The CPU energy consumption consists of the device idle energy consumption and the energy consumed during the task execution, which can be calculated by the following formula:

$$E_{i,\text{cpu}} = E_{i,\max} \cdot \mu_{i,\text{cpu}} + E_{i,\text{idle}} \tag{10}$$

where $E_{i,\max}$ is the maximum energy consumption, $\mu_{i,\text{cpu}}$ is the CPU utilization rate, and $E_{i,\text{idle}}$ is the energy consumption in idle state.

The energy consumption for wireless transmission is assumed to consume energy, p_i, per thousand transmit bytes and then the energy consumption can be written by

$$E_{i,\text{trans}} = \left(d_{i,\text{in}} + d_{i,\text{out}}\right) \cdot p_i. \tag{11}$$

Therefore, the energy consumption of user i at time slot t is calculated by

$$\begin{aligned} E_i = {} & x_1 \cdot E_{i,\text{cpu}}^{\text{loc}} + x_2 \cdot \left(E_{i,\text{cpu}}^{\text{LEO}} + E_{i,\text{trans}}^{\text{LEO}}\right) \\ & + x_3 \cdot \left(E_{i,\text{cpu}}^{MEO} + E_{i,\text{trans}}^{MEO}\right) \\ & + x_4 \cdot \left(E_{i,\text{cpu}}^{\text{GEO}} + E_{i,\text{trans}}^{\text{GEO}}\right) \\ & \sum_{i=1}^{4} x_i = 1. \end{aligned} \tag{12}$$

Thus, the total energy consumption of the system is given by

$$E = \sum_{i=1}^{|U|} E_i \tag{13}$$

where $|U|$ represents the total number of users.

D. Problem Formulation

We aim to minimize the system energy consumption while ensuring the service quality and user experience. The task delay can be incorporated as a constraint for the objective optimization. Specifically, for the latency-sensitive tasks, if the execution delay exceeds its maximum allowable threshold, it is considered as an offloading failure, which adversely affects the overall system performance and user satisfaction. Based on this, we define the following objective function:

$$\min \quad \sum_{t=1}^{T} E(t) \tag{14}$$

$$\text{s.t.} \quad \sum_{i=1}^{4} x_i = 1 \qquad \forall x_i \in \{0, 1\} \tag{15}$$

$$0 < D_i < D_{\max,i} \qquad \forall i \in U \tag{16}$$

$$0 < E_{i,\text{cpu}} \qquad \forall i \in \{U, \mathcal{M}, \mathcal{N}, \mathcal{G}\} \tag{17}$$

$$0 < E_{i,\text{trans}} \qquad \forall i \in \{\mathcal{M}, \mathcal{N}, \mathcal{G}\} \tag{18}$$

where (15) ensures that each user's task must be strictly executed locally or offloaded to one of the satellite layers: LEO, MEO, or GEO. This constraint ensures the uniqueness of task allocation and the certainty of the execution path. Constraint (16) aims to ensure that the delay of any task does not exceed its maximum allowable delay. This constraint is crucial for meeting the real-time requirements and ensuring the user experience, particularly in latency-sensitive applications. Further, (17) and (18) ensure that the energy consumption during the task execution and data transmission processes is reasonably accounted for, respectively.

Due to the integration of discrete decision, $\boldsymbol{x}$, and continuous variables, such as computational resources and bandwidth, the problem in (14) is a mixed-integer nonlinear programming (MINLP) problem. Arising from the necessity to simultaneously optimize the conflicting objectives, such as minimizing energy consumption and latency under the dynamic and

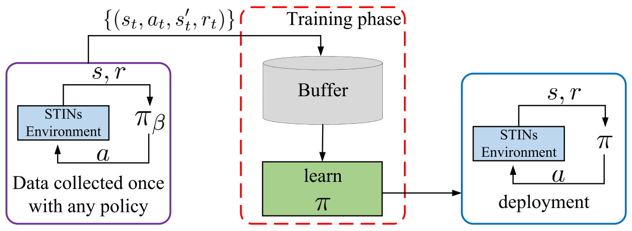


Fig. 3. Offline DRL.

uncertain network conditions, it is characterized as NP-hard. To address this issue, we will propose a solution method based on DRL as follows.

III. Offline DRL Solution

DRL requires extensive interactions with the environment to learn the optimal strategies which can lead to excessive consumption of satellite resources and potential damage to equipment while the traditional optimization algorithms avoid these learning characteristics without direct environmental interactions. So, in this section, we propose a novel offline DRL-based offloading optimization solution to learn the optimal decision policies by leveraging the historical data from traditional optimization algorithms, as shown in Fig. 3. The offline DRL uses an existing experience pool $\{(s_t, a_t, s'_t, r_t)\}$ to train the decision-making strategies without direct environmental interaction, reducing the satellite resource consumption and associated risks [17]. s_t and s'_t are the states of STINs, a_t represents the offloading decision, and r_t is the reward under the offloading decision a_t.

The offline DRL process includes three steps: first, an arbitrary policy π_β generates data stored in an experience pool; second, an enhanced offline reinforcement algorithm learns directly from this data; and finally, the trained policy π is deployed for validation and application. Although the offline DRL initially collects the experience data once, the strategy can be iteratively optimized for better performance.[1]

We also adopt the discrete SAC algorithm with an offline policy for the task offloading decisions. The RL algorithms are used to collect the preliminary experience data due to their table-based characteristics, which involve less interaction with the environment. However, these methods struggle with the large state space in satellite task offloading. Inspired by [11], we discretize the satellite system state into four levels: 1) low; 2) medium; 3) busy; and 4) high. The tasks are similarly categorized based on the computational needs and data volume. Specifically, the categories for the state parameters are set as follows.

1) *Task MIPS Requirements:* "low" for the tasks requiring fewer than 20 000 MIPS, "medium" for those requiring between 20 000 and 100 000 MIPS, and "high" for the tasks exceeding 100 000 MIPS.
2) *Maximum Allowed Latency:* "low" for the tasks that can tolerate delays of up to 6 s, "medium" for those that can withstand latency between 6 and 15 s, and "high" for the tasks needing to be executed in less than 15 s.
3) *Current Ground User MIPS Usage:* "low" for the usage under 30 000 MIPS, "medium" for usage between 30 000 and 130 000 MIPS, and "high" for the usage exceeding 130 000 MIPS.
4) *Ground User CPU Utilization:* "low" for the utilization below 25%, "medium" for 25%–50%, "busy" for 50%–75%, and "high" for 75%–100%.
5) *Satellite CPU Utilization:* Following the same categorization as the ground user CPU utilization, the CPU usage of a satellite is classified into "low" (below 25%), "medium" (25%–50%), "busy" (50%–75%), and "high" (75%–100%).

To meet the neural networks' input requirements in the discrete SAC algorithm, we use one-hot encoding for these categorical state features, transforming them into a format suitable for neural networks. The following sections detail the implementation of the discrete SAC algorithm under an offline policy. Similar to [11], the reward r_t is defined as

$$\hat{r}_t = \epsilon E + D \tag{19}$$

where ϵ is the parameter that balances task delay and energy consumption. Note that, in order to reduce system latency and energy consumption, we choose the action a_t in the RL algorithm that minimizes $\hat{r}_t$.

A. SAC-Based Offline-DRL Solution Algorithm for Task Offloading

This section describes the adapting process of the continuous SAC algorithm into a discrete version for the task offloading decisions. We first review the continuous SAC algorithm principles and then describe the discrete SAC implementation.

The continuous SAC algorithm maximizes an objective function that includes a maximum entropy term [22], [23]. The optimal policy, denoted by π^*, is determined by maximizing the following expression:

$$\pi^* = \arg\max_{\pi} \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\tau_\pi}\left[\gamma^t(r(s_t, a_t)) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right] \tag{20}$$

where π represents the decision-making strategy and T stands for the total number of time steps. The function $r = -\hat{r} \in \mathbb{R}$ represents the reward function, with $\gamma \in [0, 1]$ indicating the discount rate. The state at each time step t is given by $s_t \in S$, and the action taken at time step t is denoted by $a_t \in A$. The term τ_π describes the distribution of trajectories induced by the policy π. α is the temperature parameter adjusting the influence of the entropy $\mathcal{H}(\pi(\cdot|s_t))$ of the policy π at state s_t, which is calculated as $\mathcal{H}(\pi(\cdot|s_t)) = -\log\pi(\cdot|s_t)$, reflecting the uncertainty of the policy at state s_t.

The goal is refined via soft policy iteration (SPI), a process that alternates between evaluating and improving policies. During the evaluation phase, the value of the current policy π is calculated using the soft state value function as

$$V(s_t) := \mathbb{E}_{a_t\sim\pi}\left[Q(s_t, a_t) - \alpha\log(\pi(a_t|s_t))\right]. \tag{21}$$

[1]In this article, we collect the data only once, rather than continuously collecting the data and iterating the algorithm. Continuous data collection and algorithm iteration are applied in the engineering projects.

In the continuous settings, the soft Q-function $Q_\theta(s_t, a_t)$ is modeled by a neural network characterized by parameters θ, and it is optimized to reduce the soft Bellman residual as

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\Big(Q_\theta(s_t, a_t) - \Big(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p(s_t,a_t)}\big[V_{\bar{\theta}}(s_{t+1})\big]\Big)\Big)^2\right] \quad (22)$$

where D represents a replay buffer that stores past experience tuples and $V_{\bar{\theta}}(s_{t+1})$ is computed using a target neural network to approximate future state values.

During the policy improvement, the policy is updated to maximize the expected return, using the soft Q-function from policy evaluation. The updated policy is constrained within a parametric distribution family, like a Gaussian distribution. The policy update process can be described by the following formula:

$$\pi_{\text{new}} = \arg\min_{\pi\in\Pi} D_{\text{KL}}\left(\pi(\cdot|s_t) || \frac{\exp\left(\frac{1}{\alpha}Q^{\pi_{\text{old}}(s_t,.)}\right)}{Z^{\pi_{\text{old}}}(s_t)}\right) \quad (23)$$

where $Z^{\pi_{\text{old}}}(s_t)$ is the partition function. In practice, this policy $\pi_\phi(a_t|s_t)$ is parameterized by neural networks with parameters ϕ and learns policy gradients by minimizing the expected Kullback–Leibler divergence

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D}\left[\mathbb{E}_{a_t\sim\pi_\phi}\left[\alpha \log\left(\pi_\phi(a_t|s_t)\right) - Q_\theta(s_t, a_t)\right]\right]. \quad (24)$$

A reparameterization trick is used to handle the expected operation of the policy output distribution. This method combines the network outputs with noise from a standard normal distribution

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D,\epsilon_t\sim N}\left[\alpha \log\left(\pi_\phi\left(f_\phi(\epsilon_t; s_t)\right) - Q_\theta\left(s_t, f_\phi(\epsilon_t; s_t)\right)\right)\right] \quad (25)$$

where π_ϕ is implicitly defined in terms of f_ϕ. $a_t = f_\phi(\epsilon_t; s_t)$, where $\epsilon \sim N(0, I)$.

Moreover, Haarnoja et al. [23] proposed a method for adaptively learning the temperature parameter, which avoids setting it as a hyperparameter. They derived the final goal for the temperature parameter, expressed as

$$J(\alpha) = \mathbb{E}_{a_t\sim\pi_t}\left[-\alpha\left(\log \pi_t(a_t|s_t) + \bar{H}\right]\right. \quad (26)$$

where $\bar{H}$ represents a fixed vector mirroring the hyperparameter that dictates the desired level of entropy in the policy. Since it involves expected operations, directly minimizing the above expression poses challenges, so a Monte Carlo method based on samples from the replay buffer is used for estimation.

We extend the continuous action SAC algorithm to a discrete action version for the offline DRL [24]. The key modification changes the policy $\pi_\phi(a_t|s_t)$ to output the discrete action probabilities, transforming the Q-function from $Q \in \mathbb{R}$ to $Q \in \mathbb{R}^{|A|}$. The policy function also changes to output a valid probability distribution of actions $\pi \in [0, 1]^{|A|}$. This allows the direct calculation of the expected action values without the reparameterization trick.

Previously, to minimize the objective function $J_\pi(\phi)$, the soft state value functions are typically estimated using Monte Carlo methods with samples from the replay buffer. This estimation involves the computing expectations of the action probability distribution. Under our discrete action framework, since the action space is discrete and we can directly obtain the probability distribution for each action, the expected value of the actions can be directly calculated, allowing for (21) to be rewritten as

$$V(s_t) := \pi(s_t)^T\left[Q(s_t, a_t) - \alpha \log(\pi(s_t))\right]. \quad (27)$$

Similarly, the temperature objective function (26) is rewritten as

$$J(\alpha) = \pi_t(s_t)^T\left[-\alpha\left(\log \pi_t(s_t) + \bar{H}\right]\right.. \quad (28)$$

To minimize (24), we use the reparameterization trick, resulting in (25). Now, as the policy outputs an explicit action distribution, we can directly compute the expectation without relying on the reparameterization trick. Therefore, (25) can be rewritten as

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D}\left[\pi_t(s_t)^T\left[\alpha \log\left(\pi_\phi(s_t)\right) - Q_\theta(s_t)\right]\right]. \quad (29)$$

Algorithm 1 SAC-Based Offline DRL Approach for Task Offloading in STINs

```
 1: Initialize $Q_\theta \in \mathbb{R}^{|A|}$, $\pi_\phi \in [0, 1]^{|A|}$
 2: ▷ Initialize an empty replay buffer
 3: $\mathcal{D} \leftarrow \emptyset$
 4: ▷ Data collected onece with RL
 5: $\mathcal{D} \leftarrow \{(s_t, a_t, r(s_t, a_t))\}$
 6: for each epoch do
 7:     for each batch do
 8:         ▷ Selecte a batch data
 9:         $\mathcal{D}_b \leftarrow \{s_b, a_b, r_b\}$
10:         ▷ Compute the Q-function
11:         $Q(s_b, a_b)$
12:         ▷ Compute the policy network
13:         $\pi_\phi(s_b)$
14:         ▷ Compute the temperature network
15:         $\pi_t(s_b)$
16:         ▷ Update the Q-function weights
17:         $\theta \leftarrow \theta - \lambda\nabla_\theta J_Q(\theta)$
18:         ▷ Update the policy network weights
19:         $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$
20:         ▷ Update the temperature network weights
21:         $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha J(\alpha)$
22:     end for
23: end for
24: Output: $\phi$
```

In short, the proposed approach is summarized in Algorithm 1. Initially, we use the RL algorithms [11] to interact with a satellite–ground integrated network and collect an experience data set $\mathcal{D}$. We then use the discretized SAC algorithm to learn from these historical experience data $\mathcal{D}$. The SAC training process follows the conventional deep learning workflow. First, a batch of data is randomly selected from data set $\mathcal{D}$ to compute the Q-function, policy network, and temperature network. Next, loss functions are calculated according to (22), (28), and (29). The parameters of the Q-function, policy network, and temperature network are then updated through stochastic gradient descent. Finally, we deploy the trained policy network in our satellite–ground integrated network to assess the strategy's practical performance.

B. Algorithm Convergence

Similar to [30], we analyze the convergence of the proposed algorithm.

The proposed SAC algorithm is based on the principle of maximum entropy RL, and the soft Bellman equation for the state value function (27) is defined as follows.

$$V^*(s_t) = \max_{\pi} \mathbb{E}_{a_t\sim\pi(\cdot|s_t)}\big[Q(s_t, a_t) - \alpha \log \pi(a_t|s_t)\big]. \quad (30)$$

We have the following.

Lemma 1: For all $(s, a) \in \mathcal{S}\times\mathcal{A}$, the optimal value function $V^*(s)$ and the optimal policy $\pi^*(a|s)$ satisfy

$$V^*(s_t) = \alpha \log \sum_{a_t} \exp\left(\frac{Q(s_t, a_t)}{\alpha}\right) \quad (31)$$

$$\pi^*(a_t|s_t) = \frac{\exp\left(\frac{Q(s_t,a_t)}{\alpha}\right)}{\sum_{a_t} \exp\left(\frac{Q(s_t,a_t)}{\alpha}\right)}. \quad (32)$$

Proof of Lemma 1: The proof of this lemma is presented in Appendix A. ■

Next, the $Q^*(s, a)$ is defined as follows.

Definition 1: For $V^*(s)$ that satisfies

$$Q^*(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim p}\big[V^*(s')\big]. \quad (33)$$

Thus, we have

$$Q(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim p}\big[V(s')\big] \le Q^*(s, a). \quad (34)$$

We can now present the main theorem of convergence.

Theorem 1: For any initial policy π_0 and corresponding action-value function Q^{π_0}, the SPI algorithm converges to a unique fixed point such that $Q^{\pi^*}(s, a) = Q^*(s, a)$, $V^{\pi^*}(s) = V^*(s)$, and $\pi^*(a|s)$ is the optimal policy that satisfies (31) and (32).

Proof of Theorem 1: The proof of this theorem is presented in Appendix B. ■

C. Time Complexity Analysis

The time complexity of the proposed algorithm primarily arises from the SAC deployment stage, where we use a deep neural network (DNN) as the policy network. The inference time complexity of the DNN is $O(L \cdot M^2)$, where L is the number of layers in the neural network and M is the number of neurons per layer, which is critical for real-time operations. Although our focus has been on inference due to its relevance in deployment, it is also important to consider the training time complexity. The training process, conducted offline on a dedicated server using the stored historical data, has a time complexity of $O(E \cdot N \cdot L \cdot M^2)$, where E is the number of training epochs and N is the size of the training data set. Since the training is performed offline, the computational resources and time required do not impose constraints on the deployment scenario, allowing us to avoid the high resource consumption and potential risks associated with real-time interactions in the STIN environment.

IV. Performance Evaluation

This section offers simulation results showcasing the efficiency of our proposed algorithm within STINs. The task offloading algorithm and neural network are implemented with Pytorch,[2] and the system model is created using SatEdgeSim.[3]

TABLE I
System Parameters

Parameter	Value	Parameter	Value
$\delta^2_{m,n}$	7.9e-13	$\delta^2_{i,m}$	7.9e-13
$\delta^2_{n,g}$	7.9e-13	c_{mips}	$[1.5e6, 6e6, 2e7]$
$d_{i,in}$	$[500, 1.5e5, 1e6]$	$d_{i,out}$	$[500, 1000, 1e6]$
ϵ	0.01	-	-

TABLE II
Parameters of SAC

Parameter	Value	Description
γ	0.99	Discount factor
λ_π	0.0003	The learning rate of the policy network
λ	0.0003	The learning rate of the Q-function network
λ_α	0.0003	The learning rate of the temperature network
-	50	Epoch
-	64	Batch size
α	1	Initial temperature

A. Simulation Setup

We implement the proposed framework on a Linux workstation with 64-bit Ubuntu 22.04.1. The hardware for training all DRL baselines included an Nvidia GeForce RTX 3090Ti GPU with 24-GB memory and an Intel Core i9-10980XE processor with 18 cores at 3.00 GHz.

The main experimental parameters follow [16]. The energy consumption per bit of transmitted data is set at 5×10^{-8} units of energy, and the bandwidth between users and LEO satellites is 1000 Mb/s. Similarly, the bandwidth between LEO, MEO, and GEO satellites is also 1000 Mb/s. CPU resource allocation uses the traditional space-sharing algorithm, while bandwidth allocation employs a fair-sharing algorithm. Other parameters are with reference to the setting of computing and communication in STINs [16], [25], the main parameters in our system are set as in Table I. The hyperparameters set during the SAC training are shown in Table II.

Note that, in our implementations, we only normalize the rewards [28] because the states are one-hot encoded.

We assess the effectiveness of our proposed method by contrasting it with the following benchmark approaches.[4]

1) *Distance-Only [15]:* The distance delay between the user and the satellite is used as a feature to select the computing node.
2) *WEIGHT_GREEDY:* This algorithm is described in [16].
3) *RL*: A strategy from [11] that simplifies state representation by discretizing the multidimensional state space into categorical features. We use this algorithm to gain the historical decision data.
4) *Behavioral Cloning (BC):* This is a supervised learning approach that learns the mapping between states and

[2]https://pytorch.org/

[3]https://github.com/wjy491156866/SatEdgeSim

[4]In offline DRL, the goal is to determine whether the performance of the optimal policy learned from a historical data set can approach that of the algorithm which generated the historical data set [17]. DQN and DDQN are chosen as baselines because they are off-policy RL algorithms.

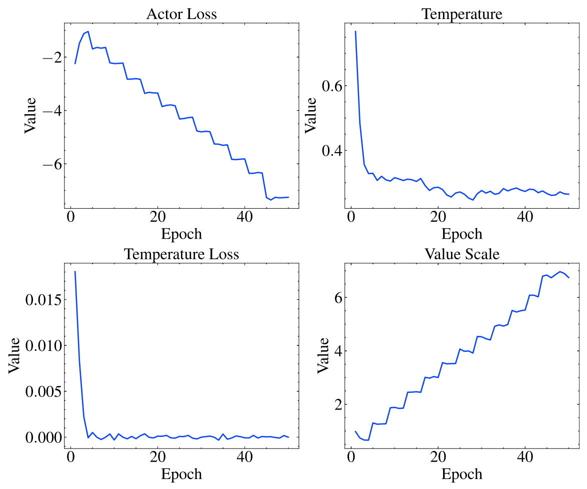


Fig. 4. Performance metrics of the proposed SAC.

actions directly from the historical experience pool using traditional deep learning models.

5) *DQN:* An offline policy DRL algorithm that begins by collecting interaction experiences with the environment, storing them in an experience pool, and then extracting sample data for training the Q network.

6) *DDQN:* To address potential overestimation issues during DQN training, DDQN introduces two Q networks to enhance learning stability.

B. Performance Evaluation

Fig. 4 presents the convergence behavior of the proposed SAC algorithm, highlighting four key metrics over 50 epochs of training. The Actor Loss plot (top-left) shows a steady decrease, indicating continuous improvement in the policy's performance. The Temperature plot (top-right) demonstrates how the exploration parameter stabilizes early in the training process, contributing to more consistent policy updates. Similarly, the Temperature Loss plot (bottom-left) rapidly converges to near-zero values, suggesting that the temperature parameter is effectively tuned. Finally, the Value Scale plot (bottom-right) exhibits a consistent upward trend, reflecting the increasing value estimation as the algorithm learns more about the optimal policy. Collectively, these results illustrate the SAC algorithm's stable and efficient convergence properties.

Fig. 5 shows the task success rates under different task offloading strategies. The results indicate that even when trained solely on the historical data from the RL algorithm, the DQN, DDQN, and SAC algorithms significantly outperform the original RL algorithm, validating the efficacy of offline DRL. The BC algorithm, trained using a supervised learning paradigm based on RL historical data, shows the improved task success rates but still falls below DQN, DDQN, and SAC, further highlighting the superiority of offline DRL with these advanced algorithms. Additionally, Fig. 5 shows that the DQN and SAC algorithms achieve the comparable task success rates with the Distance-Only algorithm, demonstrating that DQN and SAC can achieve high task success rates solely

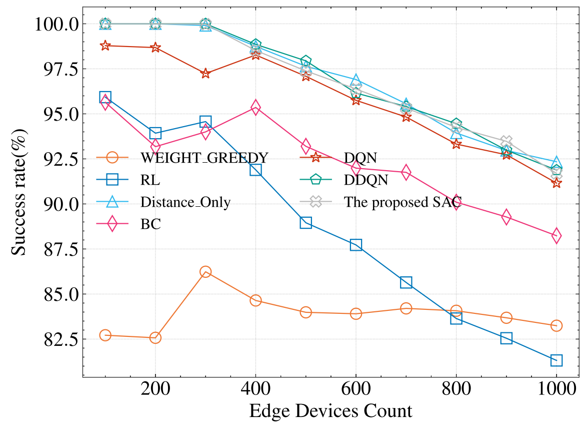


Fig. 5. Task success rate.

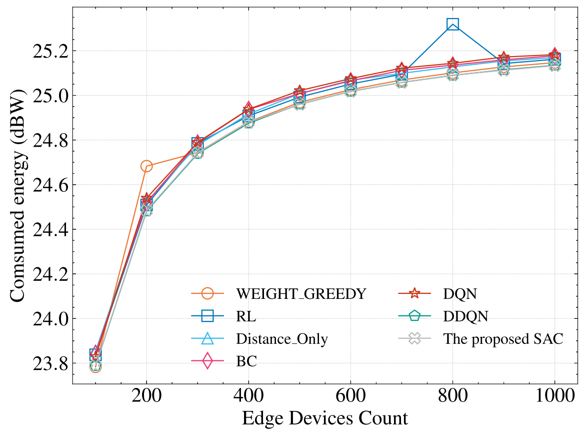


Fig. 6. Average energy consumption for individual satellites.

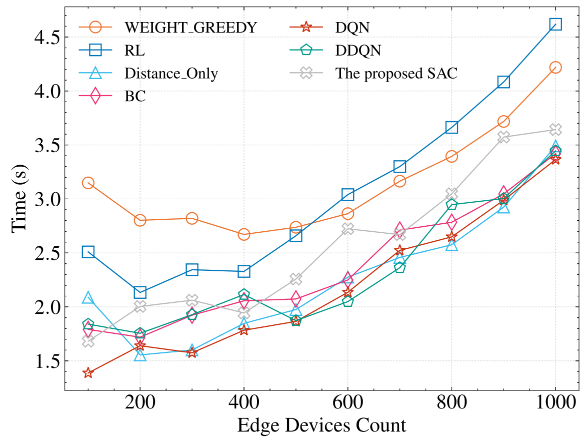
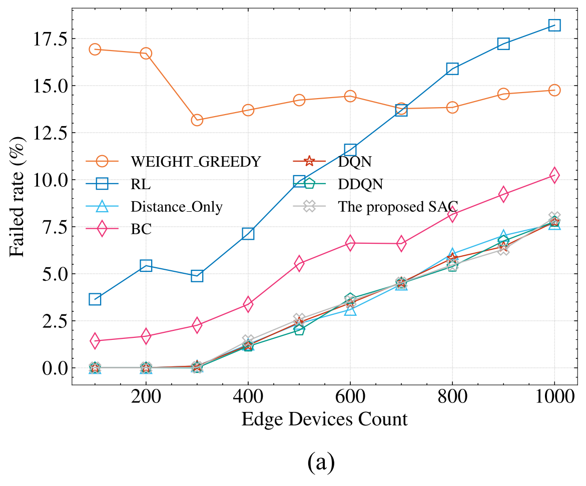


Fig. 7. Task average end-to-end delay.

based on the historical RL data. Note that the Distance-Only algorithm is currently the optimal algorithm in the SatEdgeSim simulation framework.

Fig. 6 demonstrates that with an increasing in the number of edge devices, there is a corresponding rising in the average energy consumption per satellite node. The data in Fig. 6 show that the DDQN and SAC algorithms have the lowest average energy consumption per satellite node, reaffirming the energy efficiency advantages of offline RL.



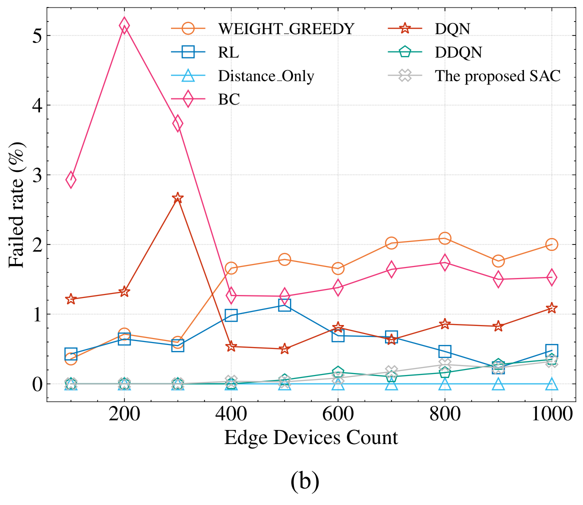
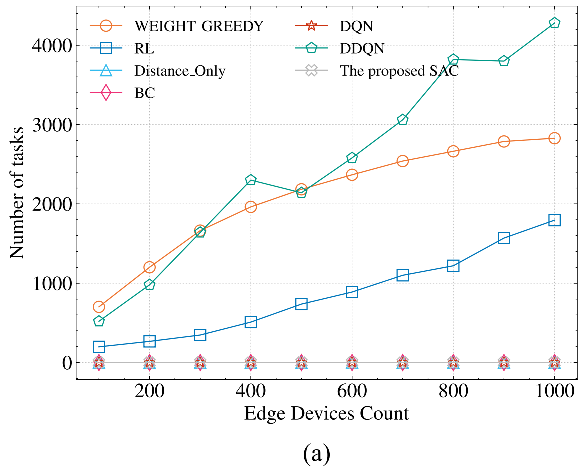
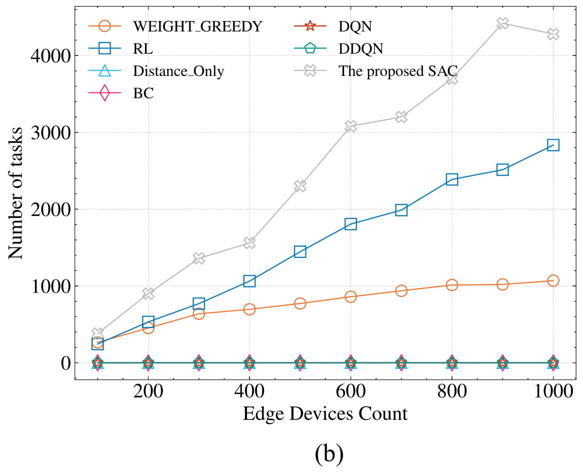


Fig. 8. Rate of task failures. (a) Task failure due to delay. (b) Task failure due to mobility.





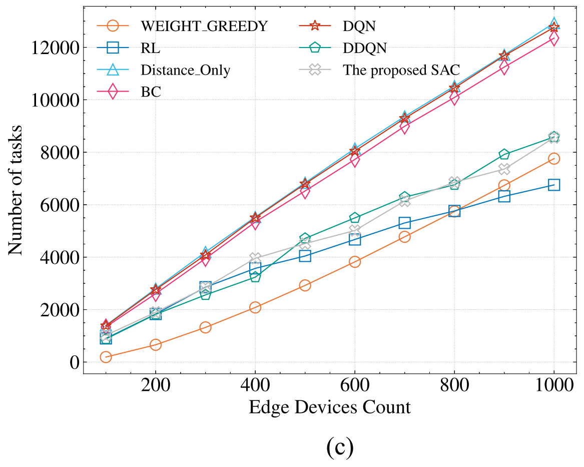


Fig. 9. Task processing counts by resource type. (a) GEO. (b) MEO. (c) LEO.

Fig. 7 analyzes the system performance from the perspective of average end-to-end delay of tasks. The DQN, DDQN, and SAC algorithms all outperform the RL baseline model, confirming the effectiveness of offline RL. The SAC algorithm exhibits a higher average end-to-end delay compared to the BC, DQN, and DDQN algorithms because it offloads more tasks to the MEO layer, whereas BC, DQN, and DDQN mainly offload tasks to the LEO layer.

Fig. 8(a) and (b) depicts the rates of task failures attributed to latency and mobility issues. The algorithms DDQN, SAC, and Distance-Only are shown to significantly mitigate these task failure rates associated with latency and mobility

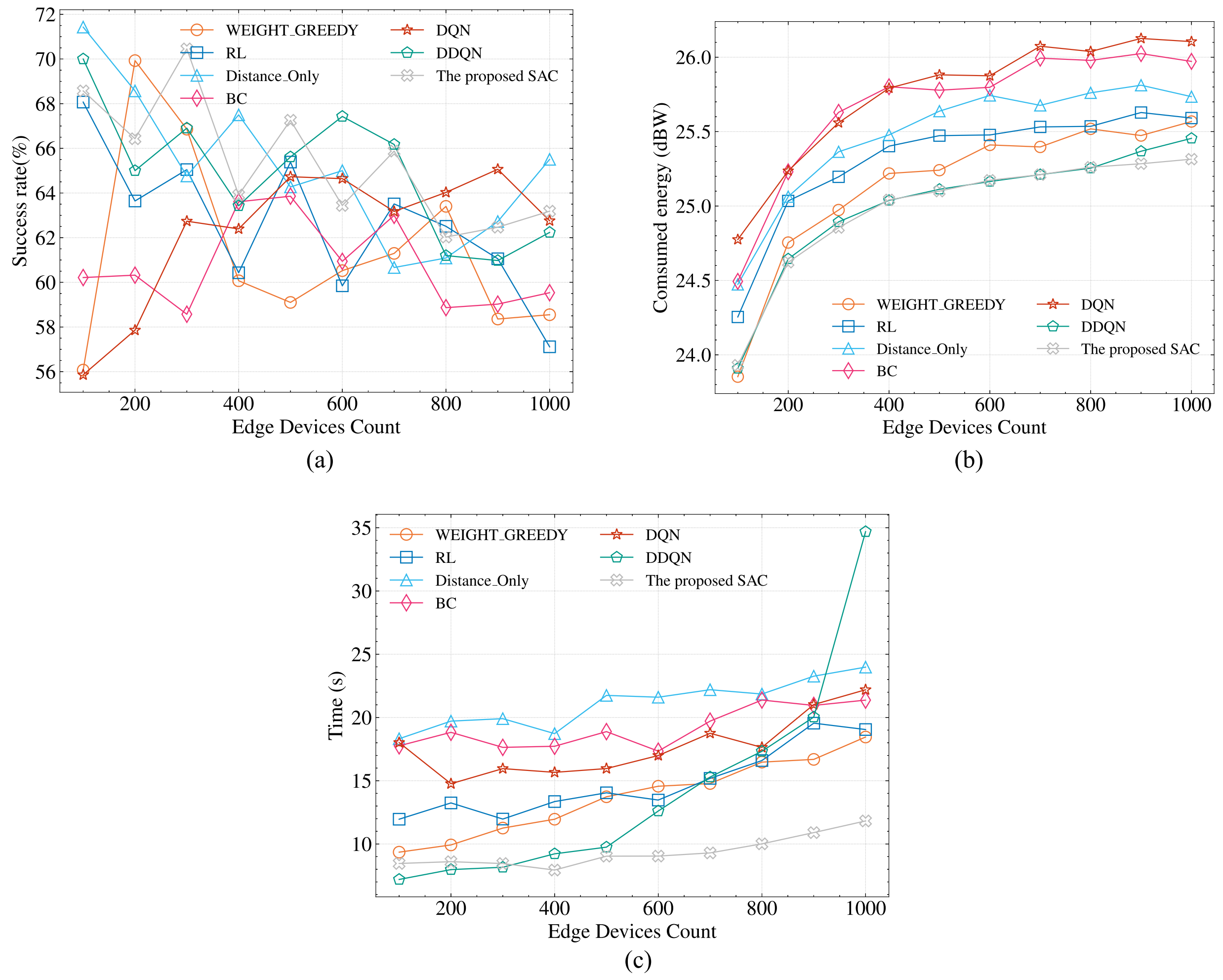


Fig. 10. Results of algorithmic robustness in diverse metrics. (a) Task success rate. (b) Average energy consumption of each satellite. (c) Task average end-to-end delay.

challenges. The Distance-Only algorithm demonstrates strong resilience against the increasing numbers of edge devices, indicating good stability against task failures caused by mobility. However, as the number of edge devices increases, task failures due to mobility issues also increase for the DDQN and SAC algorithms.

Fig. 9(a)–(c) depicts the distribution of task executions under different resource types. The figures show that the SAC algorithm tends to offload tasks to the LEO and MEO layers, whereas the DDQN algorithm favors the GEO and LEO layers. Although DDQN and SAC are derived from the same training data set and their offloading decisions are distinctly different, both algorithms display similar performance levels. This indicates that multiple effective task offloading strategies exist within satellite–ground integrated networks. Specifically, the Distance-Only algorithm prefers to offload tasks to the nearest LEO layer, primarily because it bases its offloading decisions on the proximity between users and satellites.

C. Experimental Validation of Algorithm Robustness

To further validate the robustness of the proposed SAC algorithm, this article conducts a series of tests by adjusting task parameters. Specifically, the input and output data sizes and the required MIPS computational resources for each task are increased tenfold, while all other system parameters remain unchanged. To ensure the fairness of the experiments, the BC, DQN, DDQN, and SAC algorithms are all deployed using models that has been previously trained.

Fig. 10(a) shows the performance of the Distance-Only algorithm when handling single tasks with high computational and data demands, which does not reach the excellent performance shown in Fig. 5. This suggests that algorithms based on distance as a decision criterion may not be suitable for handling single large-scale tasks. At the same time, the DDQN and SAC algorithms demonstrate comparable task success rates, both significantly outperforming the Distance-Only algorithm, fully reflecting the superiority of these two algorithms.

Fig. 10(b) and (c) presents comparisons of both the average energy consumption per satellite node and the average task end-to-end delay across different algorithms. The SAC algorithm outperforms the other algorithms on both metrics, particularly in terms of average energy consumption per satellite node and average end-to-end delay of tasks. This further verifies the robustness of the SAC algorithm, significantly superior to the BC, DQN, and DDQN algorithms.

From the above, we has experimentally verified the effectiveness and practicality of offline DRL in the task offloading scenario of STINS. The proposed SAC-based task offloading algorithm has demonstrated robustness performance when faced with computationally and data-intensive tasks, significantly outperforming the DQN and DDQN algorithms. This provides a new approach for the task offloading strategies in complex environments in the future.

V. Conclusion

This article optimizes the task offloading problem in STINs by adopting the offline DRL. It significantly reduces the direct interactions with the environments, effectively lowers the satellite resource consumption, and enhances the task processing efficiency. Experimental results demonstrate the substantial advantages in terms of latency, energy consumption, and overall system performance, clearly showcasing the potential of offline DRL in optimizing the satellite network tasks.

Future work will explore the integration of digital twin technology to further optimize the task offloading strategies and enhance the adaptability of our approach. Additionally, we will consider expanding the model to support larger-scale networks and test more complex network scenarios and conditions.

Appendix A
Proof of Lemma 1

Our goal is to maximize the state value function $V(s_t)$

$$V^*(s_t) = \max_{\pi} \mathbb{E}_{a_t\sim\pi(\cdot|s_t)}\big[Q(s_t, a_t) - \alpha \log \pi(a_t|s_t)\big]. \quad (35)$$

To introduce the Lagrange multiplier method, we need to consider the probability distribution of the policy $\pi(a|s)$, i.e.,

$$\sum_{a_t} \pi(a_t|s_t) = 1. \quad (36)$$

We construct the Lagrangian function $\mathcal{L}(\pi, \lambda)$ to represent this constraint by

$$\mathcal{L}(\pi, \lambda) = \sum_{a_t} \pi(a_t|s_t)\big[Q(s_t, a_t) - \alpha \log \pi(a_t|s_t)\big] + \lambda\left(1 - \sum_{a_t} \pi(a_t|s_t)\right) \quad (37)$$

where λ is the Lagrange multiplier.

To solve for the optimal policy $\pi^*(a_t|s_t)$, we need to take the derivative of the Lagrangian function $\mathcal{L}(\pi, \lambda)$ with respect to $\pi(a_t|s_t)$ and set the derivative equal to zero

$$\frac{\partial \mathcal{L}}{\partial \pi(a_t|s_t)} = Q(s_t, a_t) - \alpha(\log \pi(a_t|s_t) + 1) - \lambda = 0. \quad (38)$$

Then, we solve this equation to obtain $\pi(a_t|s_t)$

$$\pi(a_t|s_t) = \exp\left(\frac{Q(s_t, a_t)}{\alpha}\right) \cdot \exp\left(\frac{-\lambda - \alpha}{\alpha}\right). \quad (39)$$

Since $\pi(a_t|s_t)$ is a probability distribution, we determine λ by normalizing $\sum_{a_t} \pi(a_t|s_t) = 1$

$$\sum_{a_t} \pi(a_t|s_t) = \sum_{a_t} \exp\left(\frac{Q(s_t, a_t)}{\alpha}\right) \cdot \exp\left(\frac{-\lambda - \alpha}{\alpha}\right) = 1. \quad (40)$$

We have

$$\exp\left(\frac{-\lambda - \alpha}{\alpha}\right) = \frac{1}{\sum_{a_t} \exp\left(\frac{Q(s_t,a_t)}{\alpha}\right)}. \quad (41)$$

Therefore

$$\pi^*(a_t|s_t) = \frac{\exp\left(\frac{Q(s_t,a_t)}{\alpha}\right)}{\sum_{a_t} \exp\left(\frac{Q(s_t,a_t)}{\alpha}\right)}. \quad (42)$$

By substituting the optimal policy $\pi^*(a_t|s_t)$ into the state value function in (35), since $\pi^*(a_t|s_t)$ has already maximized this expression, we can directly use the maximized value

$$V^*(s_t) = \alpha \log \sum_{a_t} \exp\left(\frac{Q(s_t, a_t)}{\alpha}\right). \quad (43)$$

Appendix B
Proof of Theorem 1

The soft Bellman operator $\mathcal{T}^\pi$ for a given policy π is defined by

$$\mathcal{T}^\pi Q(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim p(s'|s,a)}\big[V^\pi(s')\big] \quad (44)$$

where

$$V^\pi(s) = \alpha \log \sum_{a} \exp\left(\frac{Q(s, a)}{\alpha}\right). \quad (45)$$

This operator maps a current estimate of the action-value function $Q(s, a)$ to a new estimate based on the expected rewards and future value estimates.

To prove the convergence, we need to show that the soft Bellman operator is a contraction mapping. That is, for any two action-value functions Q_1 and Q_2 [31]

$$\|\mathcal{T}^\pi Q_1 - \mathcal{T}^\pi Q_2\|_\infty \le \gamma \|Q_1 - Q_2\|_\infty. \quad (46)$$

Given that $0 \le \gamma < 1$, the contraction mapping property implies that the repeated application of $\mathcal{T}^\pi$ will converge to a unique fixed point Q^π, which satisfies the Bellman equation

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim p(s'|s,a)}\big[V^\pi(s')\big]. \quad (47)$$

Since the soft Bellman operator is a contraction, the repeated application of it under the SPI algorithm will converge to the optimal action-value function $Q^*(s, a)$. The corresponding optimal value function $V^*(s, a)$ is then

$$V^*(s) = \alpha \log \sum_{a} \exp\left(\frac{Q^*(s, a)}{\alpha}\right). \quad (48)$$

Given the convergence of $Q^*(s, a)$, the optimal policy $\pi^*(a|s)$ derived from the softmax distribution over $Q^*(s, a)$ also converges

$$\pi^*(a|s) = \frac{\exp\left(\frac{Q^*(s,a)}{\alpha}\right)}{\sum_a \exp\left(\frac{Q^*(s,a)}{\alpha}\right)}. \quad (49)$$

Finally, we demonstrate that the fixed point (Q^*, π^*) is unique. Given the contraction property of the soft Bellman operator, the fixed point to which it converges must be unique. Therefore, the algorithm converges to the unique optimal action-value function Q^* and the corresponding optimal policy π^*.

References

[1] X.-T. Li, S. Xu, Z.-P. Zhao, Z.-Y. Li, D.-A. Li, and J.-M. Zhao, "A survey on computing offloading in satellite–terrestrial integrated edge computing networks," in *Proc. ICCSN*, Shenyang, China, 2023, pp. 172–182.

[2] Y. Zhang, C. Chen, L. Liu, D. Lan, H. Jiang, and S. Wan, "Aerial edge computing on orbit: A task offloading and allocation scheme," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 1, pp. 275–285, Jan./Feb. 2023.

[3] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Satellite–terrestrial integrated edge computing networks: Architecture, challenges, and open issues," *IEEE Netw.*, vol. 34, no. 3, pp. 224–231, May/Jun. 2020.

[4] X. Zhang et al.,"Energy-efficient computation peer offloading in satellite edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 3077–3091, Apr. 2024.

[5] J. Zhou, Q. Yang, L. Zhao, H. Dai, and F. Xiao, "Mobility-aware computation offloading in satellite edge computing networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 10, pp. 9135–9149, Oct. 2024.

[6] C. Ding, J.-B. Wang, M. Cheng, M. Lin, and J. Cheng, "Dynamic transmission and computation resource optimization for dense LEO satellite assisted mobile-edge computing," *IEEE Trans. Commun.*, vol. 71, no. 5, pp. 3087–3102, May 2023.

[7] F. Chai, Q. Zhang, H. Yao, X. Xin, R. Gao, and M. Guizani, "Joint multi-task offloading and resource allocation for mobile edge computing systems in satellite IoT," *IEEE Trans. Veh. Technol.*, vol. 72, no. 6, pp. 7783–7795, Jun. 2023.

[8] H. Zhang, R. Liu, A. Kaushik, and X. Gao, "Satellite edge computing with collaborative computation offloading: An intelligent deep deterministic policy gradient approach," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 9092–9107, May 2023.

[9] Z. Ji, S. Wu, and C. Jiang, "Cooperative multi-agent deep reinforcement learning for computation offloading in digital twin satellite edge networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3414–3429, Nov. 2023.

[10] H. Guo, X. Zhou, J. Wang, J. Liu, and A. Benslimane, "Intelligent task offloading and resource allocation in digital twin based aerial computing networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3095–3110, Oct. 2023.

[11] A. Robles-Enciso and A. F. Skarmeta, "A multi-layer guided reinforcement learning-based tasks offloading in edge computing," *Comput. Netw.*, vol. 220, Jan. 2023, Art. no. 109476.

[12] S. Zhang, A. Liu, C. Han, X. Liang, X. Xu, and G. Wang, "Multiagent reinforcement learning-based orbital edge offloading in SAGIN supporting Internet of Remote Things," *IEEE Internet Things J.*, vol. 10, no. 23, pp. 20472–20483, Dec. 2023.

[13] M. Bouet and V. Conan, "Mobile edge computing resources optimization: A GEO-clustering approach," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 2, pp. 787–796, Jun. 2018.

[14] T. Leng, P. Duan, D. Hu, G. Cui, and W. Wang, "Cooperative user association and resource allocation for task offloading in hybrid GEO-LEO satellite networks," *Int. J. Satell. Commun. Netw.*, vol. 40, no. 3, pp. 230–243, 2022.

[15] M. Babaghayou, N. Chaib, L. Maglaras, Y. Yigit, and M. A. Ferrag, "Distance-only task orchestration algorithm for energy efficiency in satellite-based mist computing," 2023, *arXiv:2311.14308*.

[16] J. Wei, S. Cao, S. Pan, J. Han, L. Yan, and L. Zhang, "SatEdgeSim: A toolkit for modeling and simulation of performance evaluation in satellite edge computing environments," in *Proc. ICCSN*, Chongqing, China, 2020, pp. 307–313.

[17] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020, *arXiv:2005.01643*.

[18] M. Yan, L. Zhang, W. Jiang, C. A. Chan, A. F. Gygax, and A. Nirmalathas, "Energy consumption modeling and optimization of UAV-assisted MEC networks using deep reinforcement learning," *IEEE Sensors J.*, vol. 24, no. 8, pp. 13629–13639, Apr. 2024.

[19] Z. Wu, Z. Jia, X. Pang, and S. Zhao, "Deep reinforcement learning-based task offloading and load balancing for vehicular edge computing," *Electronics*, vol. 13, p. 1511, Apr. 2024.

[20] E. Kim, I. P. Roberts, P. A. Iannucci, and J. G. Andrews, "Downlink analysis of LEO multi-beam satellite communication in shadowed Rician channels," in *Proc. GLOBECOM*, Madrid, Spain, 2021, pp. 1–6.

[21] A. U. Chaudhry and H. Yanikomeroglu, "Laser intersatellite links in a starlink constellation: A classification and analysis," *IEEE Veh. Technol. Mag.*, vol. 16, no. 2, pp. 48–56, Jun. 2021.

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor–critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018, *arXiv:1801.01290*.

[23] T. Haarnoja et al., "Soft actor–critic algorithms and applications," 2018, *arXiv:1812.05905*.

[24] P. Christodoulou, "Soft actor–critic for discrete action settings," 2019, *arXiv:1910.07207*.

[25] X. Cao et al., "Edge-assisted multi-layer offloading optimization of LEO satellite–terrestrial integrated networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 381–398, Feb. 2023.

[26] M. Matracia, N. Saeed, M. A. Kishk, and M.-S. Alouini, "Post-disaster communications: Enabling technologies, architectures, and open challenges," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 1177–1205, 2022.

[27] Z. Liu, Y. Jiang, and J. Rong, "Resource allocation strategy for satellite edge computing based on task dependency," *Appl. Sci.*, vol. 13, no. 18, 2023, Art. no. 10027.

[28] T. Deng et al., "Entropy normalization SAC-based task offloading for UAV-assisted mobile edge computing," *IEEE Internet Things J.*, vol. 11, no. 15, pp. 26220–26233, Aug. 2024.

[29] H. Wu, X. Yang, and Z. Bu, "Deep reinforcement learning for computation offloading and resource allocation in satellite–terrestrial integrated networks," in *Proc. IEEE 95th Veh. Technol. Conf.*, Helsinki, Finland, 2022, pp. 1–5.

[30] J. Ma, "The point to which soft actor–critic converges," 2023, *arXiv:2303.01240*.

[31] B. Zhang, F. Xiao, and L. Wu, "Offline reinforcement learning for asynchronous task offloading in mobile edge computing," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 1, pp. 939–952, Feb. 2024.

Bo Xie received the M.S. degree in computer technology from Guizhou University, Guiyang, China, in 2022. He is currently pursuing the Ph.D. degree in electronic science and technology with South China Normal University, Foshan, China.

His current research interests include mobile-edge computing and optimization, and edge intelligence.

Haixia Cui (Senior Member, IEEE) received the M.S. and Ph.D. degrees in communication engineering from the South China University of Technology, Guangzhou, China, in 2005 and 2011, respectively.

She is currently a Full Professor with the School of Electronics and Information Engineering, South China Normal University, Foshan, China. From July 2014 to July 2015, she was an Advanced Visiting Scholar (Visiting Associate Professor) with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada. She has authored or co-authored more than 70 refereed journal and conference papers and one book. She also holds about 20 patents. Her current research interests are in the areas of mobile-edge computing, vehicular networks, cooperative communication, wireless resource allocation, 5G/6G, multiple access control, and power control in wireless networks.

Peng Cao received the Ph.D. degree in communication engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2001.

His current research interests include the design of short-range wireless communication systems and wearable IoT.

Yejun He (Senior Member, IEEE) received the Ph.D. degree in information and communication engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005.

From 2005 to 2006, he was a Research Associate with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong. From 2006 to 2007, he was a Research Associate with the Department of Electronic Engineering, Faculty of Engineering, The Chinese University of Hong Kong, Hong Kong. In 2012, he joined the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, as a Visiting Professor. From 2013 to 2015, he was an Advanced Visiting Scholar (Visiting Professor) with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. From 2023 to 2024, he is an Advanced Research Scholar (Visiting Professor) with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. Since 2006, he has been a faculty of Shenzhen University, Shenzhen, China, where he is currently a Full Professor with the College of Electronics and Information Engineering, the Director of Sino-British Antennas and Propagation Joint Laboratory of Ministry of Science and Technology of the People's Republic of China (MOST), the Director of the Guangdong Engineering Research Center of Base Station Antennas and Propagation, and the Director of the Shenzhen Key Laboratory of Antennas and Propagation. He was selected as a Leading Talent in the "Guangdong Special Support Program" and the Shenzhen "Pengcheng Scholar" Distinguished Professor, China, in 2024 and 2020, respectively. He has authored or co-authored more than 300 refereed journal and conference papers and seven books. He holds about 20 patents. His research interests include wireless communications, antennas, and radio frequency.

Dr. He was also a recipient of the Shenzhen Overseas High-Caliber Personnel Level B (Peacock Plan Award B) and the Shenzhen High-Level Professional Talent (Local Leading Talent). He received the Second Prize of Shenzhen Science and Technology Progress Award in 2017, the Three Prize of Guangdong Provincial Science and Technology Progress Award in 2018, the Second Prize of Guangdong Provincial Science and Technology Progress Award in 2023, and the 10th Guangdong Provincial Patent Excellence Award in 2023. He is currently the Chair of IEEE Antennas and Propagation Society-Shenzhen Chapter and obtained the 2022 IEEE APS Outstanding Chapter Award. He has served as a Technical Program Committee Member or a Session Chair for various conferences, including the IEEE Global Telecommunications Conference (GLOBECOM), the IEEE International Conference on Communications, the IEEE Wireless Communication Networking Conference, and the IEEE Vehicular Technology Conference. He served as the TPC Chair for IEEE ComComAp 2021 and the General Chair for IEEE ComComAp 2019. He was selected as a Board Member of the IEEE Wireless and Optical Communications Conference (WOCC). He served as the TPC Co-Chair for WOCC 2023/2022/2019/2015, APCAP 2023, UCMMT 2023, ACES-China2023, and NEMO 2020. He acted as the Publicity Chair of several international conferences, such as the IEEE PIMRC 2012. He is serving as an Executive Chair for 2024 IEEE International Workshop of Radio Frequency and Antenna Technologies. He is the Principal Investigator for over 40 current or finished research projects, including the National Natural Science Foundation of China, the Science and Technology Program of Guangdong Province, and the Science and Technology Program of Shenzhen City. He has served as a reviewer for various journals, such as the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, the IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION, the IEEE WIRELESS COMMUNICATIONS, the IEEE COMMUNICATIONS LETTERS, the *International Journal of Communication Systems*, and *Wireless Personal Communications*. He is serving as an Associate Editor for IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION, IEEE TRANSACTIONS ON MOBILE COMPUTING, *IEEE Antennas and Propagation Magazine*, IEEE ANTENNAS AND WIRELESS PROPAGATION LETTERS, *International Journal of Communication Systems*, *China Communications*, and *ZTE Communications*. He is a Fellow of IET and a Senior Member of the China Institute of Communications and the China Institute of Electronics.

Mohsen Guizani (Fellow, IEEE) received the B.S. (with Distinction), M.S., and Ph.D. degrees in electrical and computer engineering from Syracuse University, Syracuse, NY, USA in 1985, 1987, and 1990, respectively.

He is currently a Professor of Machine Learning with Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. Previously, he worked in different institutions in the USA. He is the author of 11 books, more than 1000 publications and several U.S. patents. His research interests include applied machine learning and artificial intelligence, smart city, Internet of Things, intelligent autonomous systems, and cybersecurity.

Dr. Guizani has won several research awards, including the 2015 IEEE Communications Society Best Survey Paper Award, the Best ComSoc Journal Paper Award in 2021, as well as five Best Paper Awards from ICC and Globecom Conferences. He was listed as a Clarivate Analytics Highly Cited Researcher in Computer Science in 2019–2022. He is also the recipient of the 2017 IEEE Communications Society Wireless Technical Committee Recognition Award, the 2018 AdHoc Technical Committee Recognition Award, and the 2019 IEEE Communications and Information Security Technical Recognition (CISTC) Award. He served as the Editor-in-Chief for IEEE NETWORK and is currently serving on the editorial board of many IEEE transactions and magazines. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker and is currently the IEEE ComSoc Distinguished Lecturer.