Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.083956

ARTICLE

Reliable Low-Latency Task Offloading and Resource Allocation Method for Space-Air-Ground Integrated Networks

Fei Bu[1], Zheng Wang[2,3,*], Yong Pan[4], Zhaomin Wu[1], Yuchen Liang[1], Zhongshan Zhu[4] and Tengfei Tu[5]

[1]Technology and Platform Department, Beijing Municipal Big Data Center, Beijing, China
[2]College of Economics and Management, Beijing University of Technology, Beijing, China
[3]Institute of Digital Economy Innovation, Beijing Academy of Science and Technology, Beijing, China
[4]Beijing Computing Center Company Ltd., Beijing Academy of Science and Technology, Beijing, China
[5]School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, China

*Corresponding Author: Zheng Wang. Email: wangzheng2020@emails.bjut.edu.cn

Received: 14 April 2026; Accepted: 11 June 2026; Published: 15 September 2026

ABSTRACT: Space-Air-Ground Integrated Networks (SAGIN) provide a multi-layered, wide-coverage computing infrastructure for distributed urban sensing systems. However, their heterogeneity and dynamics pose unprecedented challenges for task offloading and resource allocation. Existing methods struggle to simultaneously address the complexity of cross-layer decision-making and reliability assurance under uncertain conditions. This paper proposes a novel framework, termed DRL-RA, which synergistically integrates Deep Reinforcement Learning (DRL) with reliability-aware optimization. The framework consists of two complementary components: (1) a Dueling Double Deep Q-Network (D3QN) module that learns adaptive policies to make offloading decisions among various options including local execution, terrestrial edge, UAVs, and satellites; (2) a Reliability-Aware Multi-Objective Optimization Framework (RA-MOOF) that introduces explicit reliability guarantees through cross-layer link reliability modeling, node availability estimation, and smooth reliability proxy functions. Addressing the heterogeneous communication characteristics of the SAGIN architecture, this paper establishes a complete cross-layer delay model and composite reliability metrics. The reliability formulation is defined under explicitly stated conditional-independence assumptions, and the proposed smooth constraint terms are treated as surrogate CMDP costs rather than exact hard chance-constraint guarantees. Extensive experiments in a SAGIN simulation environment demonstrate that the proposed method improves the task completion rate by 3.8%, reduces average latency by 11.1%, and increases system reliability by 3.9% compared to state-of-the-art benchmarks. The optimization-only RA-Opt baseline is used as a non-real-time optimization reference for assessing reliability-aware offloading decision quality, while deployment-time decision-latency comparisons are interpreted primarily among learned inference policies. Comprehensive ablation studies and statistical validation across multiple random seeds confirm the contributions of each component, while cross-layer offloading decision analysis verifies the effectiveness of the method across different network layer selections.

KEYWORDS: Task offloading; resource allocation; deep reinforcement learning; reliability optimization; space-air-ground integrated network; urban sensing; edge computing; multi-objective optimization

1 Introduction

1.1 Research Background and Motivation

As the core network architecture for the 6G era, Space-Air-Ground Integrated Networks (SAGIN) are profoundly transforming the deployment methods and service capabilities of distributed urban sensing

Copyright © 2026 The Authors. Published by Tech Science Press. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

systems [1]. By integrating Low Earth Orbit (LEO) satellite constellations in the space layer, Unmanned Aerial Vehicle (UAV) swarms in the air layer, and terrestrial edge servers in the ground layer, SAGIN constructs a multi-layered, wide-coverage, and highly reliable computing infrastructure. This provides unprecedented support capabilities for applications such as environmental monitoring, traffic management, public safety, and emergency response in smart cities [2].

Recent studies have also moved toward graph-based DRL, multi-agent coordination, and constrained/safe RL for SAGIN and LEO edge computing [3–7]. These works reinforce the need for topology-aware and constraint-aware resource management; however, many of them emphasize representation, coordination, or performance-oriented offloading, whereas our focus is to make task-level reliability cost, redundancy, and low-latency inference jointly explicit under a unified offloading formulation.

However, the heterogeneity and dynamics of SAGIN also present unprecedented challenges for task offloading and resource allocation. Unlike the traditional "terminal-edge-cloud" three-tier architecture, each layer of SAGIN possesses unique computing capacity, communication characteristics, and reliability features: satellites offer wide-area coverage but suffer from high communication latency and weather-dependent link reliability; UAVs feature flexible deployment but are constrained by battery life and dynamic positional changes; terrestrial edge servers provide strong computing power but have a limited coverage range. How to make optimal task offloading decisions among these heterogeneous resources while ensuring end-to-end reliability has become a critical problem to be solved.

1.2 Challenges Faced by Existing Methods

Existing research on SAGIN task offloading and resource allocation can be broadly categorized into three paradigms, each exhibiting significant limitations when applied to space-air-ground integrated scenarios with strict reliability requirements.

Heuristic Methods: Such as nearest node selection and load-balancing round-robin, though computationally lightweight, these methods offer the advantage of computational simplicity. However, these methods are typically based on static rules and fail to adapt to the dynamic characteristics of SAGIN—orbital satellite motion, changing UAV positions, and fluctuating link conditions all require decision methods to possess real-time adaptive capabilities. For instance, simply selecting the nearest edge server might work well on the ground but could lead to task interruption when a satellite visibility window closes.

Optimization Methods: These methods model task offloading as mathematical programming problems, utilizing techniques like Lyapunov optimization [8], game theory [9], and convex optimization [10]. While these approaches can provide theoretical optimality guarantees under certain assumptions, accurately modeling SAGIN's cross-layer decision space and dynamic constraints is extremely difficult. Satellite orbital constraints, UAV battery constraints, and cross-layer link reliability constraints are mutually coupled, forming high-dimensional non-convex optimization problems that traditional methods struggle to solve within a limited timeframe.

Learning Methods: Deep Reinforcement Learning (DRL) has gained widespread attention for its ability to learn optimal policies through environmental interaction [11]. However, several key limitations remain unresolved:

- **Cross-Layer Decision Complexity:** The action space in SAGIN includes local execution, multiple terrestrial edge servers, multiple UAVs, and multiple satellites. The action space is highly dimensional, and the computation latency of each option varies significantly (ranging from millisecond-level edge execution to hundreds of milliseconds for satellite execution). Traditional DRL methods struggle to effectively learn such a complex decision space.

- **Heterogeneous Link Reliability:** Communication links across different layers in SAGIN exhibit distinct reliability characteristics. Terrestrial links suffer from multi-path fading, air-to-ground links are affected by Line-of-Sight (LoS)/Non-Line-of-Sight (NLoS) switching, and satellite-to-ground links are impacted by rain attenuation. Existing methods often employ simplified reliability models that fail to accurately characterize these heterogeneous features.
- **Dynamic Resource Availability:** Satellite visibility windows are limited, UAV batteries continuously deplete, and edge server loads change dynamically. These time-varying constraints make traditional static resource allocation methods difficult to apply.
- **Limited Theoretical Support:** Convergence and feasibility guarantees for DRL methods addressing SAGIN cross-layer constrained optimization problems are often not established, leading to unstable performance during actual deployment.

1.3 Main Contributions

To address the aforementioned challenges, this paper proposes a novel framework named DRL-RA (Deep Reinforcement Learning and Reliability-Aware), which synergistically integrates a Dueling Double Deep Q-Network (D3QN) with a Reliability-Aware Multi-Objective Optimization Framework (RA-MOOF), specifically designed for task offloading and resource allocation in Space-Air-Ground Integrated Networks. The main contributions of this paper are summarized as follows:

1. We establish a complete SAGIN task offloading system model, encompassing a cross-layer task offloading decision space, heterogeneous communication models (device-to-edge, device-to-UAV, device-to-satellite, and UAV-to-satellite links), hierarchical computation models, and cross-layer reliability metrics. This model accurately captures the heterogeneity and dynamics of SAGIN.
2. We propose a cross-layer reliability measurement method tailored for SAGIN, comprehensively considering delay reliability, node availability (satellite visibility windows, UAV battery constraints, server failure models), and link reliability (differential modeling for terrestrial, air-to-ground, and satellite-to-ground links). We introduce a smooth reliability proxy to provide dense reward signals and improve the numerical stability of Lagrangian multiplier updates.
3. We design a cross-layer decision module based on the Dueling Double Deep Q-Network. The action space covers four types of target options: local execution, terrestrial edge servers, UAVs, and satellites. The dueling network architecture and double Q-learning mechanism effectively handle the value estimation problem in high-dimensional action spaces.
4. We conduct extensive experiments in a SAGIN simulation environment, employing multi-random seed statistical validation ($n \geq 10$). Comprehensive ablation studies isolate the contributions of each component. Cross-layer offloading decision analysis verifies the effectiveness of the method in selecting among space, air, and ground layers.
5. We provide complete configuration details for the space layer (LEO satellite orbital parameters, inter-satellite links), air layer (UAV cruise parameters, battery models), and ground layer (edge servers, sensing devices) to ensure reproducibility.

To make the connection between the above challenges and the proposed design explicit, DRL-RA is organized around four corresponding mechanisms. First, the D3QN-based decision module addresses the cross-layer decision complexity by decomposing state value and action advantage for discrete choices among local, terrestrial edge, UAV, and satellite execution targets. Second, the reliability-aware modeling module separately characterizes link reliability, node availability, and delay-related reliability, and then integrates them into a path-level reliability measure for heterogeneous SAGIN links. Third, the CMDP state and action design incorporate queue states, node loads, link estimates, satellite visibility windows, and

UAV availability, so that dynamic resource availability can be reflected in each decision epoch. Finally, the smooth reliability cost, softplus penalty, and Lagrangian update provide a tractable surrogate for reliability-constrained optimization, while the ablation and constraint-budget results further verify the contribution of these components.

2 Related Work

2.1 Space-Air-Ground Integrated Networks

As the core architecture of 6G, the Space-Air-Ground Integrated Network has received extensive attention from both academia and industry in recent years. Zhang et al. [1] first systematically described the technical architecture and key challenges of 6G SAGIN, pointing out that cross-layer resource management and mobility support are core difficulties. Xiao et al. [12] provided a comprehensive survey of SAGIN for 6G, systematically analyzing the architectural evolution, key enabling technologies, and cross-layer resource management challenges across space, air, and ground segments. Liu et al. [2] proposed a hierarchical architecture design for SAGIN and analyzed the coordination mechanisms among the space, air, and ground segments.

In terms of task offloading, Zhang et al. [13] studied computation offloading issues in SAGIN but did not fully consider reliability constraints. Zhang et al. [14] proposed a deep learning-based resource allocation method for SAGIN, but assumed perfect channel state information.

Liu et al. [15] proposed an online computation offloading framework for collaborative space/aerial-aided edge computing toward 6G systems, addressing the heterogeneous resource constraints across space and aerial layers. Zhang et al. [16] investigated energy-efficient computation peer offloading in satellite edge computing networks, jointly optimizing offloading decisions and power allocation under limited on-board processing capabilities and inter-satellite coordination constraints.

Huang et al. [5] further studied joint offloading and resource allocation for hybrid cloud-edge computing in SAGINs with hybrid action spaces, while Liu et al. [6] modeled edge-computing offloading in SAGIN from a game-theoretic perspective. These studies enrich the recent offloading literature, but reliability-aware redundancy and smooth reliability-cost control remain less explored.

The rapid development of LEO satellite constellations has made on-board computing feasible. Zhang et al. [16] analyzed the edge computing potential of satellite edge computing networks and investigated energy-efficient peer offloading strategies. The main challenges for satellite computing include limited on-board resources, dynamic topologies, and restricted visibility windows [17].

UAVs have unique advantages as mobile edge computing platforms. Mozaffari et al. [18] systematically reviewed the challenges and opportunities of UAV communication and computation. UAV position optimization and trajectory design have become research hotspots [19], and Yan et al. [20] employed deep reinforcement learning to optimize task offloading in UAV-assisted Internet of Vehicles, jointly optimizing UAV trajectory and resource allocation, but these works usually assume static task loads and do not consider dynamic task arrivals and reliability constraints.

2.2 Task Offloading in Edge Computing

Task offloading in edge computing environments has been extensively studied over the past decade. The fundamental tradeoff between local execution and remote offloading has been extensively studied in mobile cloud computing and edge computing literature [21,22]. Mao et al. [8] subsequently introduced Lyapunov optimization for dynamic computation offloading, achieving near-optimal performance while

ensuring queue stability. Their method dynamically adjusts offloading decisions based on queue backlogs and channel conditions, providing theoretical bounds for average delay and energy consumption.

Game-theoretic methods are widely adopted in distributed task offloading where multiple users compete for limited edge resources. Chen [9] modeled the multi-user offloading problem as a potential game, proved the existence of Nash equilibrium, and designed a distributed algorithm to reach the equilibrium state. Building on this line of work, Chen et al. [23] proposed an efficient multi-user computation offloading framework for mobile-edge cloud computing that achieves Nash equilibrium through a distributed potential game formulation. Although game-theoretic approaches enable decentralized decision-making, they usually require an iterative convergence process that may not be suitable for delay-sensitive urban sensing applications.

Due to the battery constraints of mobile devices, energy-efficient offloading has received particular attention. Zhang et al. [10] proposed a joint optimization framework for energy-efficient task offloading under energy and latency constraints, considering both energy and deadline constraints. Their work showed that intelligent offloading decisions can significantly extend device lifetime while satisfying application requirements. However, these optimization methods usually assume known or predictable energy arrival patterns, which may not hold in practical urban sensing scenarios with unpredictable environmental conditions.

Geng et al. [11] proposed a DRL-based distributed computation offloading framework for vehicular edge computing, formulating the problem as an MDP and leveraging deep deterministic policy gradient to minimize system latency and energy consumption in a fully distributed manner. Xie et al. [24] jointly optimized computation offloading, resource allocation, and distributed edge service caching for multi-user MEC systems using DRL, addressing the interplay between offloading decisions and caching placement.

2.3 Deep Reinforcement Learning in Resource Management

The application of deep reinforcement learning in edge computing resource management has emerged as a prominent research direction. Mnih et al. [25] demonstrated the effectiveness of Deep Q-Networks (DQN) in learning complex control policies, inspiring subsequent applications to the task offloading problem. Chen et al. [26] proposed a distributed DRL offloading framework where each device maintains an independent DQN agent, learning to make offloading decisions based on local observations. Their method requires minimal coordination overhead while significantly outperforming heuristic baselines.

Cai et al. [3] introduced a graph-based deep reinforcement learning approach for dynamic resource allocation in SAGIN, modeling the heterogeneous network topology as a graph and leveraging spatial attention mechanisms to capture cross-layer dependencies. Li et al. [4] proposed a multi-agent reinforcement learning framework based on the centralized training with decentralized execution (CTDE) paradigm for computation offloading and resource allocation in LEO satellite edge computing networks, considering satellite mobility, dynamic topology, and heterogeneous resource constraints.

To address the challenge of limited training data and dynamic environments, Zhang et al. [22] proposed a digital twin-driven intelligent task offloading framework for collaborative MEC, leveraging digital twin technology to create virtual replicas of the physical system for real-time state monitoring and predictive offloading decisions. Zhu et al. [27] proposed ECO-SDIoT, a DRL-based edge computing offloading algorithm for software-defined IoT, utilizing SDN to provide global state information for making offloading decisions across edge and cloud resources.

Li et al. (2023) proposed TapFinger, utilizing MAPPO for decomposed scheduling and resource allocation, employing conflict resolution and action masking to achieve stability. Their work demonstrated

the importance of handling infeasible actions through masking, a technique related to our reliability-aware design. HIPPO-MAT [28] emphasized decentralized execution and practical routing conflict handling, reporting extensive testbed validation. MA-CDMP [29] introduced model-based planning and mean-field conditioning, providing formal bounds for constraint satisfaction. Jia et al. [30] proposed MASTD3, a multi-agent DRL framework for real-time proportional computation offloading in industrial IoT edge computing environments, addressing device selfishness and limited bandwidth challenges. Shao et al. [31] investigated DRL-based resource management for UAV-assisted MEC under jamming attacks, jointly optimizing hover point selection, power control, and computing resource allocation while adapting to dynamic interference. Our work differs by focusing on explicit reliability modeling and smooth proxies instead of hard constraints, and adopting a Centralized Training-Distributed Execution (CTDE) paradigm with reliability guarantees.

2.4 Constrained and Safe Reinforcement Learning

Constrained reinforcement learning is becoming increasingly critical for safety-critical applications. Achiam et al. [32] proposed Constrained Policy Optimization (CPO), which provides guaranteed policy improvement while satisfying cost constraints. Tessler et al. [33] introduced Reward Constrained Policy Optimization (RCPO), combining Lagrangian methods with trust region optimization. FOCOPS [34] provides faster convergence for constrained policy optimization through first-order methods.

Wachi et al. [7] provided a comprehensive survey of constraint formulations in safe reinforcement learning, systematically classifying approaches including CMDP, chance constraints, and state-wise constraints, and offering theoretical guidance for designing reliability-constrained optimization frameworks.

A key issue in constrained RL is the choice of constraint representation. Hard indicator functions lead to sparse gradients and unstable training [32]. Smooth proxy functions, such as sigmoid-based approximations, have been proposed to resolve this issue [33]. Our work adopts and extends these techniques for reliability constraints in task offloading.

2.5 Reliability in Distributed Computing Systems

Reliability engineering has a long history in distributed computing systems, tracing back to early work on fault-tolerant distributed systems [35]. Traditional approaches to improving reliability include redundancy [36], checkpointing [37], and failure detection [38]. In the context of edge computing, reliability issues have gained renewed attention due to the mission-critical nature of many IoT applications.

Recent studies have begun incorporating reliability considerations into task offloading decisions. Xue et al. [39] proposed a DRL-based joint service caching and computation offloading scheme for vehicular edge computing systems, effectively leveraging edge resources to reduce service latency through intelligent caching and offloading coordination. Wang et al. [40] proposed a partial computation offloading approach using dynamic voltage scaling for mobile edge computing, jointly optimizing the offloading ratio and computational speed to balance energy consumption and execution latency.

Li et al. [36] investigated energy-efficient reliability-aware offloading for delay-sensitive tasks in collaborative edge computing, jointly optimizing offloading decisions, power allocation, and computation resources while ensuring task completion probability. Cao et al. [35] proposed a reliability-aware personalized deployment framework for approximate computation IoT applications in serverless mobile edge computing, considering user-specific reliability requirements and computation accuracy tradeoffs to maximize QoS. However, these methods typically rely on traditional optimization techniques with high computational complexity or use hard constraints that result in training instability.

To make the above distinction explicit, Table 1 compares the relevant research families by topology scope, constraint treatment, and remaining limitation. The comparison is used to position this work as a reliability-aware SAGIN offloading framework that combines heterogeneous space–air–ground observability, smooth reliability-cost learning, and redundancy-aware decision support rather than optimizing a single terrestrial MEC layer or using only hard feasibility filters.

Table 1: Literature comparison and gap identification.

Research Family	Representative References	Topology Information	Constraint Treatment	Remaining Gap Addressed Here
Classical MEC offloading	[8,21,23]	Usually terminal-edge or edge-cloud	Queue, energy, or delay constraints	Limited modeling of space-air-ground heterogeneity and intermittent satellite links
UAV-assisted MEC	[18–20]	Air-ground geometry and UAV mobility	Trajectory, energy, and latency constraints	Reliability is often implicit and satellite visibility is not central
Satellite/LEO edge computing	[4,16,38]	Satellite or satellite-terrestrial topology	Energy, delay, and resource constraints	Cross-layer air-ground relay reliability and task-level redundancy are not jointly emphasized
Graph/MARL resource allocation in SAGIN	[3,4]	Graph-structured or multi-agent observations	Mainly performance-oriented reward constraints	Strong topology modeling exists, but explicit link-node-delay reliability composition remains less developed
Safe/constrained RL	[7,32–34]	General CMDP setting	Cost budgets, Lagrangian updates, and trust regions	These methods need domain-specific reliability costs for SAGIN task offloading

3 System Model and Problem Definition

3.1 Space-Air-Ground Integrated Network Architecture

We consider a Space-Air-Ground Integrated Network (SAGIN) architecture for distributed urban sensing systems, as shown in Fig. 1. This architecture consists of three heterogeneous layers that achieve reliable task offloading and resource allocation through cross-layer coordination:

Space Layer: Consists of a Low Earth Orbit (LEO) satellite constellation, denoted as $\mathcal{S} = \{s_1, s_2, \ldots, s_{N_S}\}$, with an orbital altitude of 500–2000 km. Each satellite $s \in \mathcal{S}$ is equipped with an on-board processing unit with computing capacity $F_s \in [1, 5]$ GHz, capable of processing compute-intensive tasks. Satellites are interconnected via Inter-Satellite Links (ISL), forming a space mesh network. The space layer provides wide-area coverage and massive data backhaul capabilities, suitable for scenarios such as large-scale environmental monitoring and disaster emergency response.

Air Layer: Consists of a Unmanned Aerial Vehicle (UAV) swarm, denoted as $\mathcal{U} = \{u_1, u_2, \ldots, u_{N_U}\}$, with a flight altitude of 100–500 m. Each UAV $u \in \mathcal{U}$ is equipped with an edge computing module with computing capacity $F_u \in [0.5, 2]$ GHz, offering flexible near-ground computing services. UAVs possess high mobility and can dynamically adjust their positions based on task demands, suitable for temporary coverage enhancement and hotspot region services. A UAV is called visible to device d when the device–UAV distance, elevation angle, and residual UAV energy jointly satisfy the communication and serviceability conditions used in Eqs. (3), (4) and (26); otherwise, the UAV is removed from the feasible action set at that decision epoch.

Ground Layer: Consists of terrestrial sensing devices and edge servers. The set of sensing devices is denoted as $\mathcal{D} = \{d_1, d_2, \ldots, d_{N_D}\}$, including heterogeneous devices such as environmental sensors, traffic cameras, and smart meters. Edge servers are deployed at base stations and roadside units, denoted as $\mathcal{E} = \{e_1, e_2, \ldots, e_{N_E}\}$, with computing capacity $F_e \in [10, 50]$ GHz, providing low-latency local computing services.

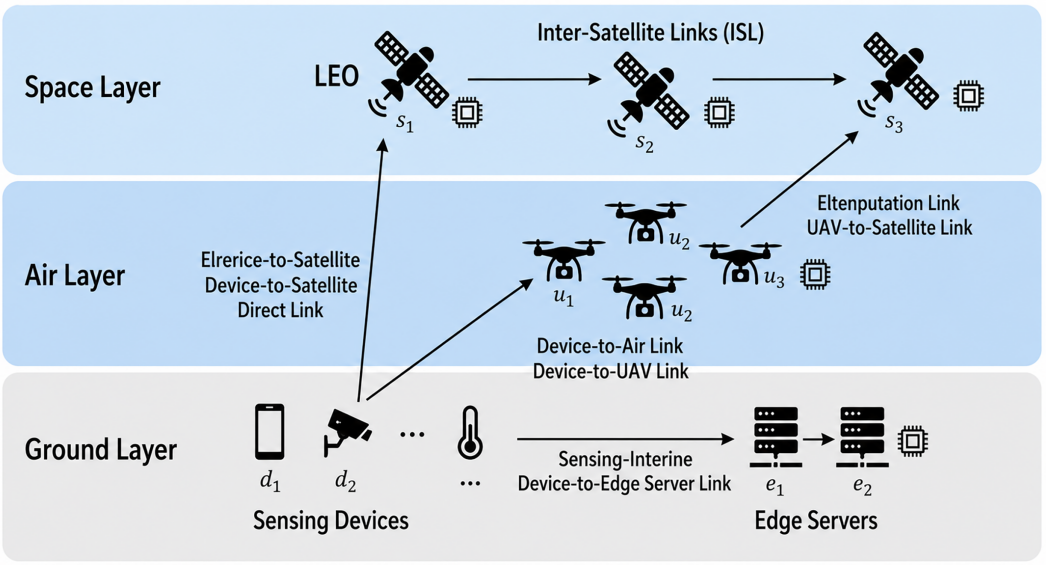

Figure 1: System architecture for SAGIN task offloading in urban sensing.

The following problem formulation is considered over a short SAGIN scheduling horizon with periodically updated observations. The LEO visible satellite set $\mathcal{V}(t)$ and the remaining visibility window are estimated from predictable ephemeris information. UAVs follow planned patrol or service trajectories and periodically broadcast their position and residual energy; sudden return or failure is reflected through the availability factor $\phi_u(t)$ and feasible-action filtering rather than through a separate emergency-control process. Edge congestion is represented by queue lengths and capacity constraints, while link states, node loads, and visibility windows are updated through beacon measurements, broadcasts, and ephemeris prediction with bounded intervals. Accordingly, rare constellation-level topology mutation, simultaneous UAV-fleet failure, or regional common-cause disruptions are not modeled as separate state-transition processes; the reliability terms in Section 3.5.3 are interpreted as conditional estimates under the observed SAGIN state.

For clarity, the considered scenario is valid under four operational assumptions: (i) LEO ephemeris and satellite visibility can be predicted within the short scheduling horizon; (ii) UAVs follow planned service trajectories and report position/energy states periodically; (iii) edge congestion is represented by

queue length and residual computing capacity rather than by irreversible server collapse; and (iv) link and node states are observable through bounded-interval updates. These assumptions are consistent with the simulation scope and do not imply robustness to arbitrary emergency topology mutation.

For readability, the main notation used in the problem formulation is summarized in Table 2.

Table 2: Key notation used in the system model.

Symbol	Meaning	Symbol	Meaning
$\mathcal{D}, \mathcal{E}, \mathcal{U}, \mathcal{S}$	Device, edge-server, UAV, and satellite sets	D_i, C_i	Input data size and required CPU cycles of task i
T_i^{max}	Deadline of task i	ρ_i^{min}	Minimum reliability requirement of task i
a_i	Offloading target selected for task i	$r_{p,q}$	Transmission rate between nodes p and q
$T_{i,n}$	End-to-end delay if task i is executed at node n	$E_{i,n}$	Energy consumption under target node n
ψ_ℓ	Reliability of link ℓ	ϕ_n	Availability of node n
$R_{i,n}^{delay}$	Smooth delay-satisfaction reliability term	$\mathcal{R}_{i,n}$	Composite reliability of assigning task i to node n
c_i^{rel}	Smooth reliability-violation cost	λ	Lagrange multiplier for the reliability cost budget
$\omega_{rel}, \omega_{pen}$	Reward weights for reliability reward and reliability penalty	$n_i^{replica}$	Number of redundant replicas for task i
Q_n	Queue backlog of node n	$F_n^{used}, F_n^{reserved}, F_n^{available}$	Used, reserved, and available computing capacity of node n
$\mathcal{V}(t)$	Visible satellite set at decision epoch t	ρ_{th}	Reliability threshold for triggering redundancy checking
d, τ_{smooth}	Expected cost budget and smoothing parameter	W	Sliding window for cost averaging and Lagrange updating

3.2 Cross-Layer Task Offloading Model

3.2.1 Task Model

Each sensing device $d \in \mathcal{D}$ generates computational tasks according to a stochastic process. A task $\tau_{d,k}$ generated by device d at time slot k is characterized by a tuple $\tau_{d,k} = (D_{d,k}, C_{d,k}, T_{d,k}^{max}, \rho_{d,k}^{min})$, where $D_{d,k}$ denotes the input data size (bits), $C_{d,k}$ denotes the required number of CPU cycles, $T_{d,k}^{max}$ denotes the maximum tolerable delay, and $\rho_{d,k}^{min}$ denotes the minimum reliability requirement.

3.2.2 Task Offloading Decisions

Under the SAGIN architecture, tasks have multiple offloading targets to choose from. We define the offloading decision variable $a_d \in \mathcal{A}$, where the action space $\mathcal{A} = \{0\} \cup \mathcal{E} \cup \mathcal{U} \cup \mathcal{S}$ includes:

- $a_d = 0$: Local execution
- $a_d = e \in \mathcal{E}$: Offload to terrestrial edge server e
- $a_d = u \in \mathcal{U}$: Offload to UAV u
- $a_d = s \in \mathcal{S}$: Offload to satellite s

Therefore, the action space size is $|\mathcal{A}| = 1 + N_E + N_U + N_S$. For a typical configuration of $N_E = 10$, $N_U = 6$, and $N_S = 3$, we have $|\mathcal{A}| = 20$.

The action a_d determines the final computation node, while the transmission path is automatically optimized by the network layer based on the current state. Let $\mathcal{P}_{d,n}$ be the set of feasible paths from device d to target node n, the system selects the optimal path that minimizes the transmission delay:

$$\mathcal{P}^*_{d,n} = \arg\min_{\mathcal{P}\in\mathcal{P}_{d,n}} \sum_{\ell\in\mathcal{P}} \frac{D_d}{r_\ell} \tag{1}$$

where r_ℓ is the transmission rate of link ℓ. For example, feasible paths from a device to a satellite include direct links (device → satellite) and relay links (device → UAV → satellite). The system automatically selects the path with the optimal delay under current conditions. This design decouples computation node decisions from routing decisions, reducing the action space dimensionality while allowing the network layer to optimize path selection using real-time channel information.

3.3 Heterogeneous Communication Models

Communication between different layers in SAGIN exhibits significant heterogeneous characteristics and needs to be modeled separately. The terrestrial path-loss configuration follows the 3GPP urban macrocell modeling family, the air-to-ground LoS probability follows the widely used UAV communication model, and the Earth-space attenuation component follows ITU-R P.618; these models provide reproducible reference assumptions but do not eliminate site-specific calibration errors in dense urban canyons or extreme rain events [18,41,42].

3.3.1 Device-to-Edge Server Link

Communication from terrestrial devices to edge servers uses cellular links, with a transmission rate of:

$$r^{g2g}_{d,e} = B_{d,e} \log_2\left(1 + \frac{P_d \cdot h_{d,e}}{N_0}\right) \tag{2}$$

where the channel gain $h_{d,e} = g_0 \cdot d^{-\alpha}_{d,e} \cdot |h_{fading}|^2$, and the path loss exponent $\alpha \in [2, 4]$. Typical transmission delay: 5–50 ms.

3.3.2 Device-to-UAV Link

Communication from devices to UAVs mainly uses Line-of-Sight (LoS) air-to-ground links. Considering UAV altitude H_u and horizontal distance $d^{hor}_{d,u}$, the elevation angle is $\theta_{d,u} = \arctan(H_u/d^{hor}_{d,u})$. The LoS link probability is:

$$P_{LoS}(\theta) = \frac{1}{1 + a_{LoS} \cdot \exp(-b_{LoS}(\theta - a_{LoS}))} \tag{3}$$

where a_{LoS}, b_{LoS} are environment-related parameters. The comprehensive transmission rate is:

$$r^{g2a}_{d,u} = B_{d,u}\left[P_{LoS} \cdot \log_2(1 + \gamma_{LoS}) + (1 - P_{LoS}) \cdot \log_2(1 + \gamma_{NLoS})\right] \tag{4}$$

where $\gamma_{LoS} = P_d \cdot |h_{LoS}|^2/N_0$ and $\gamma_{NLoS} = P_d \cdot |h_{NLoS}|^2/N_0$ are the received Signal-to-Noise Ratios (SNR) under LoS and NLoS conditions, respectively, and h_{LoS} and h_{NLoS} are the corresponding channel gains. Typical transmission delay: 10–100 ms, depending on elevation angle and environmental obstacles.

3.3.3 UAV-to-Satellite Link

The feeder link from UAVs to satellites uses the Ka-band, which is affected by atmospheric attenuation and rainfall. The transmission rate is:

$$r_{u,s}^{a2s} = B_{u,s} \log_2 \left(1 + \frac{P_u \cdot G_u \cdot G_s \cdot |h_{u,s}|^2}{N_0 \cdot L_{atm}}\right) \tag{5}$$

where G_u, G_s are the antenna gains of the UAV and satellite, respectively, and L_{atm} is the atmospheric loss factor. Typical transmission delay: 20–200 ms.

3.3.4 Device-to-Satellite Direct Link

Some tasks can be directly uploaded from devices to satellites using IoT satellite communication protocols (such as satellite versions of LoRa, NB-IoT). The transmission rate is:

$$r_{d,s}^{g2s} = B_{d,s} \cdot \eta_{mod} \cdot (1 - BER_{d,s}) \tag{6}$$

where η_{mod} is the modulation efficiency, and $BER_{d,s}$ is the Bit Error Rate. Typical transmission delay: 100–500 ms.

3.3.5 Total Cross-Layer Link Delay

For tasks offloaded across multiple layers, the total transmission delay is the sum of the delay of each link segment, and must also consider queuing delays and interference effects on multi-hop paths.

The basic transmission delay for device d passing through UAV u to satellite s is:

$$t_{d,s}^{via\_uav} = \frac{D_d}{r_{d,u}^{g2a}} + \frac{D_d}{r_{u,s}^{a2s}} \tag{7}$$

For multi-hop paths, intermediate nodes (like UAVs) need to process forwarded tasks from multiple devices, generating additional queuing delays. Let the task arrival rate when UAV u acts as a relay node be λ_u^{relay}, and the service rate be μ_u^{relay}, then the relay queuing delay is estimated using an M/M/1 model:

$$t_u^{relay\_queue} = \frac{1}{\mu_u^{relay} - \lambda_u^{relay}} \tag{8}$$

where $\mu_u^{relay} = r_{u,s}^{a2s}/\bar{D}$ is the relay service rate of the UAV-to-satellite link, and $\bar{D}$ is the average task data size.

In multi-hop paths, adjacent links may experience interference. Especially when multiple UAVs transmit to the same satellite simultaneously, multiple access interference occurs. We adopt a simplified interference model, incorporating the interference effect into an effective transmission rate:

$$r_{u,s}^{eff} = r_{u,s}^{a2s} \cdot \left(1 - \alpha_{int} \cdot \frac{N_u^{active}}{N_u^{max}}\right) \tag{9}$$

where N_u^{active} is the number of concurrently transmitting UAVs, N_u^{max} is the maximum concurrency supported by the system, and $\alpha_{int} \in [0, 0.3]$ is the interference coefficient. As N_u^{active} increases, the effective transmission rate drops, and the delay increases correspondingly.

The comprehensive total delay for a multi-hop path considering queuing and interference is:

$$T_{d,s}^{multihop} = \frac{D_d}{r_{d,u}^{g2a}} + t_u^{relay\_queue} + \frac{D_d}{r_{u,s}^{eff}} + t_s^{queue} + t_s^{comp} \tag{10}$$

The aforementioned multi-hop delay model is used in experiments to more accurately estimate the actual delay in satellite offloading scenarios. Because queuing delay and interference effects are stochastic, we use expectation values for estimation. In high-load scenarios, these factors may cause the actual delay of satellite offloading to be significantly higher than estimates considering only transmission delay.

3.4 Hierarchical Computing Models

3.4.1 Local Execution

Delay and energy consumption for local execution:

$$T_d^{local} = \frac{C_d}{f_d^{local}}, \quad E_d^{local} = \kappa_d \cdot (f_d^{local})^2 \cdot C_d \tag{11}$$

where f_d^{local} is the device CPU frequency, and κ_d is the energy consumption coefficient.

3.4.2 Remote Execution Unified Model

When a task is offloaded to a remote node $n \in \mathcal{A}\backslash\{0\}$, the total delay and energy consumption are uniformly modeled as follows.

The transmission delay using the optimal path $\mathcal{P}_{d,n}^*$:

$$t_{d,n}^{trans} = \sum_{\ell \in \mathcal{P}_{d,n}^*} \frac{D_d}{r_\ell} = \min_{\mathcal{P} \in \mathcal{P}_{d,n}} \sum_{\ell \in \mathcal{P}} \frac{D_d}{r_\ell} \tag{12}$$

The total delay is:

$$T_{d,n}^{remote} = t_{d,n}^{trans} + t_n^{queue} + t_n^{comp} + t_{n,d}^{recv} \tag{13}$$

where t_n^{queue} is the queuing delay at node n, $t_n^{comp} = C_d / f_{d,n}$ is the computation delay, and $t_{n,d}^{recv}$ is the result return delay.

The transmission energy consumption is:

$$E_{d,n}^{trans} = P_d^{tx} \cdot t_{d,n}^{trans} = P_d^{tx} \cdot \sum_{\ell \in \mathcal{P}_{d,n}^*} \frac{D_d}{r_\ell} \tag{14}$$

where P_d^{tx} is the device transmission power. For multi-hop paths, the device only needs to send data once, and transmission energy depends on the total transmission time.

Wait and receive energy consumption is:

$$E_{d,n}^{wait} = P_d^{idle} \cdot (t_n^{queue} + t_n^{comp}), \quad E_{d,n}^{recv} = P_d^{rx} \cdot t_{n,d}^{recv} \tag{15}$$

Total energy consumption is:

$$E_{d,n}^{remote} = E_{d,n}^{trans} + E_{d,n}^{wait} + E_{d,n}^{recv} \tag{16}$$

3.4.3 Edge Server Execution

When offloading to edge server e, the feasible path set is $\mathcal{P}_{d,e} = \{\text{Direct Link}\}$, and transmission delay is:

$$t_{d,e}^{trans} = \frac{D_d}{r_{d,e}^{g2g}} \tag{17}$$

Queuing delay is modeled using an M/M/1 model:

$$\mathbb{E}[t_e^{queue}] = \frac{1}{\mu_e - \lambda_e}, \quad \mu_e = \frac{F_e}{\bar{C}} \tag{18}$$

3.4.4 UAV Execution

When offloading to UAV u, the feasible path set is $\mathcal{P}_{d,u} = \{\text{Device} \rightarrow \text{UAV Direct Link}\}$, and transmission delay is:

$$t_{d,u}^{trans} = \frac{D_d}{r_{d,u}^{g2a}} \tag{19}$$

UAVs possess mobility, and their positions change over time, resulting in dynamic changes in link delay. Let the position of UAV u at time slot t be $\mathbf{p}_u(t) = [x_u(t), y_u(t), H_u]$, then the distance from device to UAV is:

$$d_{d,u}(t) = \sqrt{(x_d - x_u(t))^2 + (y_d - y_u(t))^2 + H_u^2} \tag{20}$$

3.4.5 Satellite Execution

When offloading to satellite s, the feasible path set includes both direct and relay options:

$$\mathcal{P}_{d,s} = \{\underbrace{\{d \rightarrow s\}}_{\text{Direct Path}}, \underbrace{\{d \rightarrow u \rightarrow s : u \in \mathcal{U}_{visible}\}}_{\text{Relay Path}}\} \tag{21}$$

where $\mathcal{U}_{visible}$ is the current set of visible UAVs. Optimal path transmission delay:

$$t_{d,s}^{trans} = \min\left\{\frac{D_d}{r_{d,s}^{g2s}}, \min_{u \in \mathcal{U}_{visible}}\left(\frac{D_d}{r_{d,u}^{g2a}} + \frac{D_d}{r_{u,s}^{a2s}}\right)\right\} \tag{22}$$

Satellites have limited computing capabilities but can process massive tasks in parallel, and their queuing delay follows an M/G/1 model. Additionally, satellite motion results in limited coverage windows, requiring tasks to be completed within the satellite's visibility window.

3.5 Cross-Layer Reliability Model

In SAGIN, reliability is influenced by various factors, including link quality, node availability, and cross-layer transmission risks. The composite reliability model below uses a conditional-independence approximation: after conditioning on the current observed state, residual link failures, target-node availability, and delay-deadline satisfaction are treated as separable components. This approximation is common in

reliability block diagrams and path-success modeling, but it can be optimistic when unobserved common causes, such as regional weather or shared power faults, affect multiple components simultaneously.

3.5.1 Link Reliability

The reliability modeling for various links is as follows.

Terrestrial link reliability is:

$$\psi_{d,e}^{g2g} = 1 - Q\left(\sqrt{\frac{2\gamma_{d,e}}{\gamma_{th}}}\right) \tag{23}$$

where $Q(\cdot)$ is the Q-function, and γ_{th} is the demodulation threshold SNR.

Comprehensively considering LoS and NLoS conditions, air-to-ground link reliability is:

$$\psi_{d,u}^{g2a} = P_{LoS} \cdot \psi_{LoS} + (1 - P_{LoS}) \cdot \psi_{NLoS} \tag{24}$$

where $\psi_{LoS} = 1 - Q\left(\sqrt{2\gamma_{LoS}/\gamma_{th}}\right)$ and $\psi_{NLoS} = 1 - Q\left(\sqrt{2\gamma_{NLoS}/\gamma_{th}}\right)$ are link reliabilities under LoS and NLoS conditions, respectively.

Considering rain attenuation and atmospheric scintillation, satellite-to-ground/satellite-to-air link reliability is:

$$\psi^{sat\_link} = \exp\left(-\frac{A_{rain}}{A_{th}}\right) \cdot \psi_{scint} \tag{25}$$

where A_{rain} is rain attenuation, and ψ_{scint} is the scintillation factor.

3.5.2 Node Availability

Edge server availability modeling adopts a two-state Markov model, with availability $\phi_e = \mu_e/(\lambda_e + \mu_e)$, and a failure rate $\lambda_e \in [10^{-5}, 10^{-4}]$/hour.

UAV availability modeling considers battery constraints and mechanical failures, with availability:

$$\phi_u(t) = \phi_u^{mech} \cdot \mathbb{I}\left[\frac{E_u^{remain}(t)}{E_u^{total}} > \theta_{bat}\right] \tag{26}$$

where ϕ_u^{mech} is mechanical reliability, $E_u^{remain}(t)$ is remaining power, and θ_{bat} is the minimum power threshold.

Considering on-board resource constraints and orbital switching, satellite availability is modeled as:

$$\phi_s(t) = \phi_s^{hw} \cdot \mathbb{I}[s \in \mathcal{V}(t)] \tag{27}$$

where $\mathcal{V}(t)$ is the set of visible satellites at time t.

3.5.3 Composite Reliability Metric

The overall reliability of offloading a task from device d to target node $n \in \mathcal{A}\backslash\{0\}$ is calculated using the link reliabilities on the optimal path $\mathcal{P}_{d,n}^*$:

$$\mathcal{R}_{d,n} = R_{d,n}^{delay} \cdot \phi_n \cdot \prod_{\ell \in \mathcal{P}_{d,n}^*} \psi_\ell \tag{28}$$

where $\mathcal{P}^*_{d,n}$ is the aforementioned optimal transmission path, ψ_ℓ is the reliability of link ℓ, ϕ_n is the availability of the target node, and $R^{delay}_{d,n}$ is the delay reliability proxy:

$$R^{delay}_{d,n} = \sigma\left(\frac{1 - \mathbb{E}[T_{d,n}]/T^{max}_d}{\tau_{smooth}}\right) \tag{29}$$

where $\sigma(\cdot)$ is the sigmoid function, and τ_{smooth} is the smoothing parameter. The normalized slack $1 - \mathbb{E}[T_{d,n}]/T^{max}_d$ is positive when the expected delay is below the deadline and negative otherwise. This dimensionless expression corrects the scale of the delay slack and keeps the analytical formula consistent with the reliability proxy used throughout the framework. The sigmoid is chosen because it is a differentiable relaxation of the hard event $\mathbb{I}[T_{d,n} \le T^{max}_d]$ and has a probabilistic interpretation: if the residual delay uncertainty around $\mathbb{E}[T_{d,n}]$ follows a zero-mean logistic distribution with scale proportional to $\tau_{smooth} T^{max}_d$, Eq. (29) equals the probability of meeting the deadline. A smaller τ_{smooth} makes the proxy closer to a binary indicator but increases gradient saturation, while a larger value improves smoothness but weakens the distinction between feasible and infeasible delays.

For satellite offloading, reliability calculation is uniformly based on the delay-optimal path $\mathcal{P}^*_{d,s}$. Let the intermediate node selected by the delay-optimal path be $u^* = \arg\min_{u \in \mathcal{U}_{visible}} \left(\frac{D_d}{r^{g2a}_{d,u}} + \frac{D_d}{r^{a2s}_{u,s}}\right)$, then:

$$\mathcal{R}_{d,s} = R^{delay}_{d,s} \cdot \phi_s \cdot \begin{cases} \psi^{g2s}_{d,s}, & \text{if delay-optimal path is direct} \\ \psi^{g2a}_{d,u^*} \cdot \psi^{a2s}_{u^*,s}, & \text{if delay-optimal path is via UAV relay} \end{cases} \tag{30}$$

For local execution, the link-product term is omitted and the reliability reduces to the delay component and device-side execution availability. For redundant execution, the single-path reliability above is used as the per-replica success probability in Section 4.2.3. When multiple replicas share hidden common failure sources, Eq. (30) may overestimate the true success probability; the implementation mitigates this by preferring different layers and physical locations for replicas, while a full correlated-failure model is left as future work.

3.6 Constrained Markov Decision Process Definition

We model the SAGIN task offloading problem as a Constrained Markov Decision Process (CMDP) $\langle \mathcal{S}, \mathcal{A}, P, R, C, \gamma \rangle$:

- **State Space $\mathcal{S}$:** The state vector $\mathbf{s}_{d,k}$ contains six types of information: task features, device status, status of nodes across layers, cross-layer link conditions, reliability metrics, and satellite visibility windows.
- **Action Space $\mathcal{A}$:** $a_d \in \{0\} \cup \mathcal{E} \cup \mathcal{U} \cup \mathcal{S}$, encompassing local execution and cross-layer offloading choices.
- **Transition Probability P:** Determined by task arrivals, UAV mobility, satellite orbital motion, and link dynamics.
- **Reward Function R:** $r_d = -\omega_1 \frac{T_d}{T^{max}_d} - \omega_2 \frac{E_d}{E^{max}_d}$.
- **Cost Function C:** $c_d = \text{softplus}(\rho^{min}_d - \mathcal{R}_{d,a_d})$ (smooth reliability violation).
- **Discount Factor** $\gamma \in (0,1)$.

Optimization Objective: Find a policy π^* that maximizes expected cumulative reward while satisfying the expected cumulative cost constraint:

$$\begin{aligned} \pi^* = \arg\max_{\pi} \quad & \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \\ \text{s.t.} \quad & \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c_t\right] \leq d \end{aligned} \tag{31}$$

where d is the cost budget, indicating the maximum tolerable reliability violation.

Execution Mode and Observability Assumptions

This paper adopts a single-agent Centralized Training-Distributed Execution (CTDE) paradigm. Specific assumptions are as follows.

In the training phase, a centralized trainer has access to global state information, including: (1) task queue status of all devices; (2) loads and availability of all edge servers, UAVs, and satellites; (3) channel state information for cross-layer links; (4) satellite orbital positions and visibility window predictions. This information is used to train a unified policy network $\pi(a|s;\boldsymbol{\theta})$.

In the execution phase, each sensing device d independently runs an identical copy of the policy network. The local observation $\mathbf{o}_d$ of device d is a subset of the global state $\mathbf{s}$:

$$\mathbf{o}_d = [\tau_d, \text{Local queue state}, \text{Neighbor node state}, \text{Link quality estimate}, \text{Satellite window}] \tag{32}$$

where "Neighbor node state" includes edge servers and UAVs within device d's communication range, and "Link quality estimate" is based on periodic beacon measurements.

Task characteristics τ_d and local queue states are maintained locally by the device. Edge server/UAV loads are obtained through periodic broadcasts. Satellite visibility windows are calculated via ephemeris prediction. Link qualities are estimated through beacon measurements and channel models. During execution, the observation dimensionality is approximately 30%–40% of the global state, yet key decision information (task features, target node loads, link reliability estimates) is obtainable.

Because all devices use the same policy network and a similar observation structure, decision behaviors maintain consistency. In practice, we add device positional encoding during training to enable the policy to adapt to devices at different locations.

4 Proposed DRL-RA Framework

This section details the proposed Deep Reinforcement Learning and Reliability-Aware (DRL-RA) framework. This framework consists of two collaborative components: (1) a D3QN-based task offloading decision module (Component A), and (2) a Reliability-Aware Multi-Objective Optimization Framework (Component B).

The two components are coupled as a decision-and-verification pipeline rather than as separate post-processing blocks: Fig. 2 shows how task features, node/link states, and visibility information feed the D3QN decision module and the RA-MOOF reliability evaluator, while Fig. 3 details the execution order from state observation and action selection to cost evaluation, Lagrangian update, and optional redundancy activation.

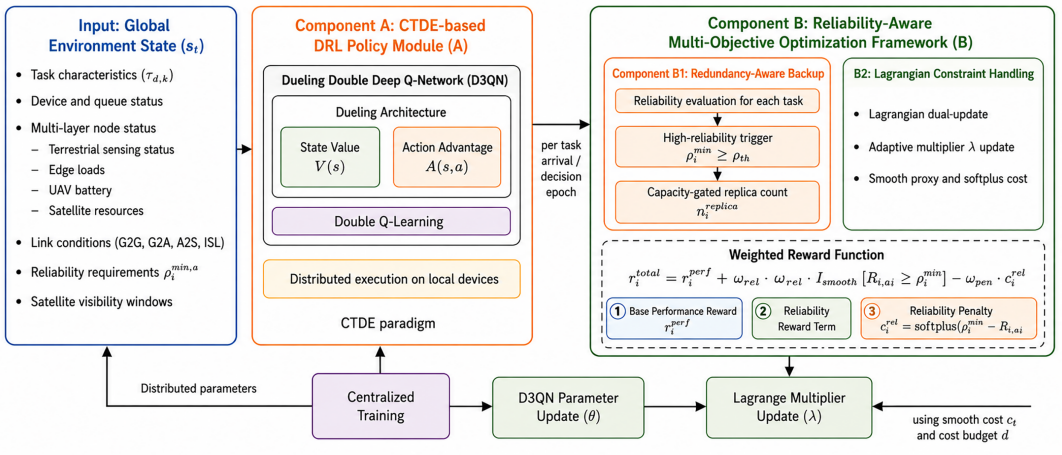

Figure 2: Overall architecture of DRL-RA. The input state includes task features, node load, link reliability estimates, and satellite/UAV visibility; the D3QN module selects a discrete offloading action, and RA-MOOF converts reliability estimates into a smooth cost, Lagrangian penalty, and optional redundancy decision. The arrows describe implemented data dependencies: D3QN is invoked once for each task-arrival decision, RA-MOOF evaluates the selected assignment at the same decision epoch, and redundancy is activated only after reliability and capacity checks.

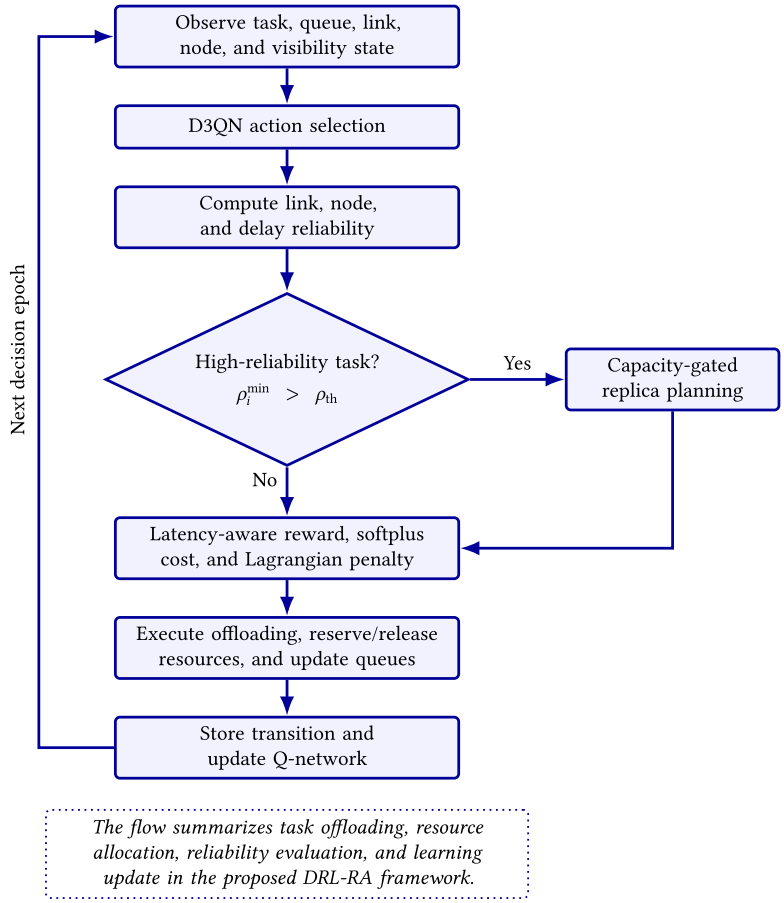

Figure 3: Methodological flowchart of the proposed DRL-RA framework. The flow summarizes state observation, D3QN-based task offloading, link/node/delay reliability evaluation, capacity-gated redundancy planning, latency-aware reward and constraint-cost construction, resource reservation/release, queue update, and Q-network learning.

4.1 Component A: D3QN-Based Decision Module

4.1.1 Dueling Network Architecture

The core of Component A is the Dueling Double Deep Q-Network (D3QN). The Q-function is decomposed as:

$$Q(s,a;\boldsymbol{\theta}) = V(s;\boldsymbol{\theta}^V) + \left(A(s,a;\boldsymbol{\theta}^A) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\boldsymbol{\theta}^A)\right) \tag{33}$$

where $V(s)$ represents the state value function, and $A(s,a)$ represents the advantage function. This architecture is particularly suitable when action values are similar across many states.

4.1.2 Double Q-Learning

To resolve overestimation bias, we employ double Q-learning:

$$y_k = r_k + \gamma Q_{target}(s_{k+1}, \arg\max_{a'} Q_{main}(s_{k+1}, a'); \boldsymbol{\theta}^-) \tag{34}$$

4.1.3 Action Space Design

Action $a_i \in \{0, 1, \ldots, M\}$ represents a categorical choice rather than fractional allocation. For categorical decisions, a DQN designed for discrete actions is appropriate; since we do not perform partial offloading, Dirichlet distributions are unnecessary.

4.2 Component B: Reliability-Aware Multi-Objective Optimization Framework

4.2.1 Smooth Reliability Proxy

A key contribution of this paper is utilizing a smooth reliability proxy instead of hard indicator functions. The reliability constraint cost is calculated as:

$$c_i^{rel} = \text{softplus}\left(\rho_i^{min} - \mathcal{R}_{i,a_i}\right) \tag{35}$$

where $\text{softplus}(x) = \log(1 + e^x)$ provides a smooth approximation of $\max(0, x)$. This surrogate does not claim exact preservation of hard per-task chance constraints. It defines a smooth CMDP cost whose expectation is optimized by the Lagrangian update. Because $\max(0, x) \leq \text{softplus}(x)$ for all x, satisfying a budget on the softplus cost is conservative with respect to the expected positive reliability deficit, but it cannot guarantee zero violation probability for every individual task.

4.2.2 Lagrangian Constraint Handling

We adopt the Lagrangian method to handle reliability constraints. The Lagrangian objective is:

$$\mathcal{L}(\pi, \lambda) = \mathbb{E}_\pi[r] - \lambda(\mathbb{E}_\pi[c] - d) \tag{36}$$

where $\lambda \geq 0$ is the Lagrangian multiplier updated via projected stochastic subgradient ascent on the dual variable:

$$\lambda_{k+1} = \max\left(0, \lambda_k + \eta_\lambda(\mathbb{E}_\pi[c] - d)\right) \tag{37}$$

4.2.3 Redundancy-Aware Backup Policy

For critical tasks where $\rho_i^{min} > \rho_{th}$, we implement redundancy:

$$n_i^{replica} = \min\left(\left\lceil \frac{\log(1-\rho_i^{min})}{\log(1-\bar{\mathcal{R}}_i)} \right\rceil, n_{max}\right) \tag{38}$$

Eq. (38) follows from the reliability of parallel independent replicas. If each replica succeeds with average probability $\bar{\mathcal{R}}_i$, the probability that all n replicas fail is $(1-\bar{\mathcal{R}}_i)^n$, and the probability that at least one replica succeeds is $1-(1-\bar{\mathcal{R}}_i)^n$. Requiring $1-(1-\bar{\mathcal{R}}_i)^n \geq \rho_i^{min}$ gives $n \geq \log(1-\rho_i^{min})/\log(1-\bar{\mathcal{R}}_i)$ because $\log(1-\bar{\mathcal{R}}_i) < 0$. Replicas are distributed across different servers to minimize correlated failures. This derivation therefore uses approximate conditional independence among replica success events rather than assuming that all physical failures in SAGIN are uncorrelated. In deployment, replicas are preferentially placed on different layers, links, or physical regions to reduce shared-risk exposure. If multiple replicas share the same weather-affected air-to-ground channel, backhaul link, or regional interference source, Eq. (38) may become optimistic; modeling such common-cause failures through correlated-failure factors or copula-based reliability terms is left as a focused extension.

After the policy selects a primary action a_i, for tasks with high reliability requirements ($\rho_i^{min} > \rho_{th}$), the system creates replicas following these steps:

1. Calculate the required number of replicas $n_i^{replica}$ according to Eq. (38), ensuring overall reliability meets the requirement.
2. Select replica targets from the available node set $\mathcal{A}\backslash\{a_i\}$, prioritizing nodes with high failure independence from the primary target node (e.g., different layers, different physical locations).
3. Reserve computing resources on each replica target node n_j:

$$F_{n_j}^{reserved} \leftarrow F_{n_j}^{reserved} + \frac{C_i}{T_i^{max} - t_{current}} \tag{39}$$

Resource reservation ensures replicas are not rejected due to insufficient capacity.

4. Replica tasks are added to the target nodes' task queues, utilizing the same priority rules as the primary task. Queue state update:

$$Q_{n_j} \leftarrow Q_{n_j} + \frac{D_i}{r_{d,n_j}} \tag{40}$$

5. The first successfully completed replica triggers a cancellation signal, terminating the execution of other replicas. The deduplication logic is implemented via tracking task IDs:

$$\text{if } \tau_i^{completed} \text{ then cancel}(\{\tau_i^{replica,k}\}_{k=1}^{n_i^{replica}}) \tag{41}$$

6. Prior to creating replicas, the system checks the available capacity of each target node:

$$F_{n_j}^{available} = F_{n_j}^{total} - F_{n_j}^{reserved} - F_{n_j}^{used} \tag{42}$$

If $F_{n_j}^{available}$ is insufficient to support the replica, that node is skipped or the replica count is reduced.

To avoid double counting, the system maintains a unified resource ledger recording the resource footprint of each task. Pre-reserved resources are immediately released upon task completion or cancellation. If all feasible edge/UAV/satellite nodes lack sufficient residual capacity, the system does not create virtual resources; it reduces the effective replica count to the number of feasible targets and records the remaining

reliability gap as a constraint cost. This capacity-gated rule prevents the redundancy mechanism from hiding resource-exhaustion cases in boundary scenarios.

4.3 Reliability-Constrained Integration of D3QN Decisions and RA-MOOF

Integration is achieved through a reliability-weighted reward function:

$$r_i^{total} = r_i^{perf} + \omega_{rel} \cdot \mathcal{R}_{i,a_i} \cdot \mathbb{I}_{smooth}[\mathcal{R}_{i,a_i} \geq \rho_i^{min}] - \omega_{pen} \cdot c_i^{rel} \tag{43}$$

where $\mathbb{I}_{smooth}$ is a smoothed indicator function using sigmoid. The weights ω_{rel} and ω_{pen} determine the reliability–latency–energy tradeoff: increasing ω_{rel} or ω_{pen} makes the policy more conservative and usually lowers constraint violations, while smaller values prioritize latency and energy. Therefore, the weights should be selected according to the service-level agreement of the target urban-sensing application rather than treated as universal constants.

Component A provides rapid categorical decisions through a neural-network forward pass, but reward shaping alone does not enforce reliability constraints. Component B provides explicit reliability evaluation and a smooth constraint-violation cost, but it is inefficient as a stand-alone online mixed-integer optimizer. Their integration uses D3QN for fast action selection and RA-MOOF for reliability-aware reward/cost construction; the resulting guarantee is a surrogate expected-cost guarantee under the CMDP assumptions rather than a hard per-task guarantee under arbitrary correlated failures.

During online deployment, the interaction frequency is one D3QN–RA-MOOF cycle per task-arrival decision epoch. First, the current task, queue, node-load, visibility, and link-state observations are encoded into $\mathbf{s}_t$. Second, D3QN outputs the primary offloading action a_t. Third, RA-MOOF evaluates link, node, and delay reliability for that selected assignment at the same epoch and forms the smooth cost c_i^{rel}. Fourth, redundancy planning is not executed for every task indiscriminately; it is triggered only when the task has a high reliability requirement or when the selected assignment leaves a positive reliability gap after capacity filtering. Finally, the task is executed, queues/resources are updated, and the Lagrange multiplier is updated over the sliding cost window W.

4.4 RA-Opt Optimization Problem Definition

RA-Opt is the pure optimization implementation of Component B and is used as an optimization reference in ablation comparisons. It solves the resource-allocation problem online at each decision epoch and is therefore not directly comparable with trained DRL policies in terms of deployment-time inference latency. In this paper, RA-Opt is positioned as a non-real-time optimization reference for assessing the quality of reliability-aware offloading decisions, rather than as a directly deployable neural-inference baseline for decision-latency comparison. Its complete optimization problem definition is as follows.

Decision variables include:

- $x_{i,n} \in \{0,1\}$: Whether task i is offloaded to node n.
- $f_{i,n} \geq 0$: CPU frequency allocated to task i (if $x_{i,n} = 1$).

Objective function where ω_1 and ω_2 weight normalized delay and normalized energy, respectively; increasing ω_1 favors lower latency, while increasing ω_2 favors lower energy consumption:

$$\min_{\mathbf{x},\mathbf{f}} \sum_{i=1}^{N} \sum_{n \in \mathcal{A}} x_{i,n} \left(\omega_1 \frac{T_{i,n}}{T_i^{max}} + \omega_2 \frac{E_{i,n}}{E_i^{max}} \right) \tag{44}$$

Constraints:

$$\begin{aligned}
&\sum_{n\in\mathcal{A}} x_{i,n} = 1, \quad \forall i \quad \text{(Each task is offloaded to exactly one node)}\\
&T_{i,n}\cdot x_{i,n} \le T_i^{max}, \quad \forall i,n \quad \text{(Delay constraint)}\\
&\mathcal{R}_{i,n} \ge \rho_i^{min} - \varepsilon_i, \quad \forall i,n : x_{i,n} = 1 \quad \text{(Reliability constraint)}\\
&\sum_{i:x_{i,n}=1} f_{i,n} \le F_n^{max}, \quad \forall n \quad \text{(Capacity constraint)}\\
&x_{i,n} \in \{0,1\}, \quad f_{i,n} \ge 0 \quad \text{(Variable domains)}
\end{aligned} \tag{45}$$

Composite reliability $\mathcal{R}_{i,n}$ is calculated according to Eq. (28), containing link reliability ψ_ℓ, node availability ϕ_n, and delay reliability proxy $R_{i,n}^{delay}$. Because $\mathcal{R}_{i,n}$ involves sigmoid functions, the problem is a non-convex Mixed Integer Programming (MIP) problem.

We employ the CVXPY framework utilizing the ECOS_BB solver for MIP processing. Relaxed problems can be approximately solved through sequential convex optimization. In experimental configurations, solver tolerance is set to 10^{-4}, with a maximum of 1000 iterations.

4.5 Complexity and Feasibility Discussion

For one decision epoch, D3QN inference requires one forward pass over $|\mathcal{A}|$ discrete actions; with a three-layer MLP of hidden widths $(256, 128, 64)$, the dominant cost is linear in the number of action scores and matrix multiplications. The Lagrangian update adds only $O(1)$ arithmetic per update window, while redundancy checking is $O(|\mathcal{A}|\log|\mathcal{A}|)$ if candidate nodes are sorted by reliability or residual capacity. By contrast, RA-Opt contains binary offloading variables and continuous CPU-allocation variables, so its worst-case mixed-integer solving cost can grow exponentially in the number of tasks and candidate nodes.

The feasibility guarantee is stated at the surrogate-CMDP level. Under bounded rewards and costs, finite action sets, and standard stochastic-approximation step-size conditions, primal-dual Lagrangian updates converge to a stationary point or neighborhood of the surrogate constrained objective in the tabular or compatible-function-approximation setting. With nonlinear neural networks, the paper does not claim global convergence; instead, feasibility is empirically checked through the expected cost $\mathbb{E}[c]$ and constraint violation rate in Table 8. This distinction is important because the softplus cost is a conservative smooth upper bound of the positive reliability deficit, not an exact hard reliability indicator.

As summarized in Algorithm 1, the proposed training procedure couples D3QN-based action selection with RA-MOOF reliability evaluation, optional replica activation, and Lagrangian multiplier adaptation, so that policy learning and reliability-constraint control are updated within the same episode loop.

Algorithm 1: DRL-RA training algorithm

Require: Cost budget d, learning rates $\eta_\theta, \eta_\lambda$, number of episodes $N_{episodes}$, max steps T_{max}
Require: Window $W = 100$, replica threshold ρ_{th}, max replicas n_{max}
Ensure: Trained policy network $Q(s, a; \boldsymbol{\theta})$
1: Initialize main network $Q(s, a; \boldsymbol{\theta})$ and target network $Q(s, a; \boldsymbol{\theta}^-)$, $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
2: Initialize experience replay buffer $\mathcal{D}$ (capacity $|\mathcal{D}|_{max} = 10^5$)
3: Initialize Lagrange multiplier $\lambda_0 \leftarrow 1.0$, cost moving average buffer $\mathcal{C} \leftarrow \varnothing$
4: Initialize ε-greedy parameters: $\varepsilon \leftarrow 1.0$, decay rate $\varepsilon_{decay} \leftarrow 0.995$, lower bound $\varepsilon_{min} \leftarrow 0.01$
5: **for** episode $ep = 1, 2, \ldots, N_{episodes}$ **do**
6: **for** $t = 1, 2, \ldots, T_{max}$ **do**

(Continued)

Algorithm 1 (continued)

Observe current state $\mathbf{s}_t$
With probability ε select a random action a_t
otherwise $a_t \leftarrow \arg\max_a Q(\mathbf{s}_t, a; \boldsymbol{\theta})$
Execute action a_t, observe immediate reward r_t and next state $\mathbf{s}_{t+1}$
Calculate composite reliability $\mathcal{R}_{t,a_t}$
Calculate constraint cost $c_t \leftarrow \text{softplus}(\rho_t^{min} - \mathcal{R}_{t,a_t})$
If $\rho_t^{min} > \rho_{th}$, calculate replica count $n_t^{replica}$, create replicas on cross-layer nodes and reserve resources
Construct composite reward $\tilde{r}_t \leftarrow r_t - \lambda_t \cdot c_t$
Store transition $(\mathbf{s}_t, a_t, \tilde{r}_t, \mathbf{s}_{t+1})$ into $\mathcal{D}$
Add c_t to cost buffer $\mathcal{C}$, if $|\mathcal{C}| > W$ remove oldest record
Sample mini-batch $\mathcal{B}$ (size 64), compute Double Q-learning targets:
$y_k \leftarrow r_k + \gamma\, Q_{target}(\mathbf{s}_{k+1}, \arg\max_{a'} Q_{main}(\mathbf{s}_{k+1}, a'); \boldsymbol{\theta}^-)$
Gradient descent to update $\boldsymbol{\theta}$:
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_\theta \nabla_{\boldsymbol{\theta}} \frac{1}{|\mathcal{B}|} \sum_{k \in \mathcal{B}} (y_k - Q(\mathbf{s}_k, a_k; \boldsymbol{\theta}))^2$
if $t \mod 100 = 0$ **then**
Calculate sliding window average cost $\bar{c} \leftarrow \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} c$
Update Lagrange multiplier: $\lambda_{t+1} \leftarrow \max\left(0,\ \lambda_t + \eta_\lambda (\bar{c} - d)\right)$
end if
end for
Periodically update target network: $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$ (every 500 steps)
Decay exploration rate: $\varepsilon \leftarrow \max(\varepsilon_{min},\ \varepsilon \cdot \varepsilon_{decay})$
end for

5 Experimental Setup

5.1 SAGIN Simulation Environment

We build a SAGIN simulation environment for urban sensing, modeling a 10 km × 10 km urban area, containing full space layer, air layer, and ground layer infrastructure.

5.1.1 Space Layer Configuration

The simulation includes $N_S = 3$ LEO satellites with an orbital altitude of 550 km (similar to Starlink orbits) and an inclination of 53 degrees. Each satellite is equipped with an on-board processing unit with a compute capacity of $F_s \in [2, 4]$ GHz and 10 GB of storage. Satellites inter-connect via laser inter-satellite links with a 10 Gbps link capacity. Satellite visibility windows last approximately 10 min, and maximum elevation angles depend on user locations.

Satellite Visibility Window Management Mechanism: Due to the movement of LEO satellites, the visibility window over a specific ground area is limited (around 10 min), heavily impacting task offloading decisions. This paper handles visibility window constraints using the following mechanisms:

(1) **Ephemeris Prediction:** The system predicts future satellite positions and visibility at time $t + \Delta t$ based on orbital parameters (altitude, inclination, right ascension of the ascending node, etc.). For the short prediction horizon used in this simulation ($\Delta t < 5$ min), the orbital-propagation model provides second-level time resolution for visibility-window estimation.

(2) **Visibility Check:** At decision time t, the system first checks target satellite s's visibility status $\mathbb{I}[s \in \mathcal{V}(t)]$, where $\mathcal{V}(t)$ is the set of visible satellites. Offloading to a satellite is only allowed if it is visible.

(3) **Remaining Visibility Time Estimation:** For currently visible satellites, the system estimates remaining visibility time T_s^{remain} and compares it with expected task completion time $T_{d,s}^{total}$. If $T_s^{remain} < T_{d,s}^{total} + T_{margin}$ (where T_{margin} is a safety margin, typically 30 s), the satellite is deemed unselectable to prevent interruption upon window closure.

(4) **Adversarial Scheduling Scenarios:** Under extreme loads or emergencies, multiple high-priority tasks might compete for limited windows. Here, the system deploys redundancy backup policies, broadcasting critical tasks to multiple visible satellites to boost reliability during window handovers. This mechanism is evaluated only as a high-load visibility-window competition scenario; it is not a claim of robustness to arbitrary constellation topology mutation, which is outside the observed-state formulation described in Section 3.

5.1.2 Air Layer Configuration

The simulation deploys $N_U = 6$ UAVs cruising at altitudes of 150–300 m and speeds of 5–15 m/s, conducting fixed-point patrol missions over the city. Each UAV carries an edge module with $F_u \in [0.5, 1.5]$ GHz capacity and a 5000 mAh battery lasting around 30 min. UAV-ground communication leverages millimeter waves with a 500 MHz bandwidth.

5.1.3 Ground Layer Configuration

The ground layer contains $N_D = 100$ sensing devices and $N_E = 10$ edge servers. Device types include environmental sensors (40%), traffic cameras (25%), smart meters (20%), and public safety sensors (15%). Edge servers are placed at cellular base stations with a 1.5 km coverage radius and capacities of $F_e \in [10, 50]$ GHz.

5.1.4 Task Generation

Tasks follow a Poisson process with sinusoidally varying intensity over time to capture daily patterns. To preserve the heterogeneity of urban sensing workloads, the task-size, CPU-cycle, deadline, and reliability-requirement ranges are sampled by device category as summarized in Table 3.

Table 3: Task characteristics by device type (sourced from references).

Type	Data (KB)	CPU (M)	Delay (s)	Reliability
Env. Sensor	50–200	100–500	10–60	0.85–0.90
Traffic Cam	500–2000	1000–5000	1–5	0.95–0.99
Smart Meter	10–50	50–200	30–120	0.90–0.95
Public Safety	200–1000	500–2000	1–3	0.98–0.995

Note: Task parameters are derived from typical urban sensing applications: environmental sensor data from real IoT smart city deployments [43,44]; traffic camera parameters aligned with video analytics tasks [21]; smart meter specs matching smart grid application standards [45]; public safety specs tracking emergency response needs [44]. The average per-device arrival rate is 12.5 tasks/s, modeling typical sensor network loads [43].

5.1.5 Cross-Layer Link Parameters

Because each offloading action may traverse terrestrial, aerial, or satellite links, the simulator assigns different bandwidth, delay, and reliability ranges to each link class; these cross-layer communication assumptions are listed in Table 4 and are used when computing transmission delay and link reliability.

Table 4: Cross-layer link parameter configurations.

Link Type	Bandwidth	Typical Delay	Reliability
Device-Edge	20 MHz	5–50 ms	0.92–0.98
Device-UAV	100 MHz	10–100 ms	0.85–0.95
Device-Sat	200 kHz	100–500 ms	0.80–0.90
UAV-Sat	500 MHz	20–200 ms	0.88–0.96
Inter-Sat Link	10 Gbps	5–20 ms	0.99

The heterogeneous link parameters in SAGIN are shown in Table 4.

Terrestrial link path loss follows the 3GPP Urban Macrocell model: $PL(d) = 128.1 + 37.6 \log_{10}(d)$ dB. Air-to-ground LoS parameters: $a = 9.61$, $b = 0.16$ (urban environment). Satellite-ground links incorporate rain attenuation via the ITU-R P.618 model.

5.1.6 Reliability Parameters

Reliability parameters across all layers:

- Edge Servers: Failure rate $\lambda_e \in [10^{-5}, 10^{-4}]$/hour, recovery rate $\mu_e \in [0.1, 0.5]$/h
- UAVs: Mechanical reliability $\phi_u^{mech} = 0.995$, battery threshold $\theta_{bat} = 20\%$
- Satellites: Hardware reliability $\phi_s^{hw} = 0.999$, availability within visibility window ≈ 0.98

5.2 Baselines and Implementation Details

We benchmark against the following methods, providing full configuration details. Note: Because our action space is discrete, methods suited purely for continuous action spaces (like DDPG) are excluded. Recent graph-DRL and MARL methods are discussed in Table 1. They are not assigned synthetic numerical scores in Table 7 because their published settings use different observation structures, agent definitions, and optimization objectives; adding unsupported numbers would reduce rather than improve fairness.

The closest state-of-the-art family to this work consists of SAGIN/LEO offloading methods based on graph DRL and MARL. These studies are discussed as closely related work because they address dynamic heterogeneous topology, but a direct numerical entry requires the observation model, agent definition, and reliability constraint to be reproducible under the same simulator. The selected numerical baselines therefore cover the comparable algorithmic families under the same task traces: heuristic offloading, queue-aware optimization, unconstrained DRL, constrained RL, the D3QN-only component, and the RA-Opt optimization reference. The reproducible settings for these baselines are given in Table 5, which makes explicit whether each method uses no constraint handling, a queue-stability rule, a policy-gradient constraint, a Lagrangian relaxation, or the proposed smooth reliability mechanism.

Table 5: Baseline method configurations.

Method	Key Parameters	Constraint Handling	Architecture
Random	None	None	None
Greedy-Nearest	Select nearest available node	None	None
Greedy-Reliability	Select most reliable node	None	None
Lyapunov	$V \in \{10, 50, 100\}$	Queue stability	Optimization
DQN	$\varepsilon \in [0.01, 1]$, decay 0.995	None	MLP [256, 128, 64]
PPO	clip $\varepsilon = 0.2$, learning rate 3×10^{-4}	None	MLP [256, 128, 64]
CPO	Trust region $\delta = 0.01$, cost limit d	Trust region	MLP [256, 128, 64]
RCPO	Same as CPO + Lagrangian	Lagrangian	MLP [256, 128, 64]
Lagrangian-PPO	$\lambda_0 = 1$, $\eta_\lambda = 0.01$	Lagrangian	MLP [256, 128, 64]
FOCOPS	Same as PPO + 1st-order constraints	1st-order method	MLP [256, 128, 64]
D3QN (Ours A)	Same as DQN + Dueling + Double	None	MLP [256, 128, 64]
RA-Opt (Ours B)	CVXPY/ECOS_BB solver	Hard constraints	Optimization
DRL-RA (Full)	Fixed values listed in Table 6	Smooth+Lagrangian	MLP [256, 128, 64]

We selected DQN and PPO as unconstrained DRL baselines because they accommodate discrete action spaces. We excluded DDPG because it is an actor-critic method custom-built for continuous action spaces, yielding deterministic continuous action values. Applying DDPG to a discrete action space necessitates discretizing actions or utilizing techniques like Gumbel-Softmax, which introduces extraneous approximation errors and diverges from the method's original design intent. Therefore, to preserve fairness and methodological rigor, DDPG is omitted. To ensure a fair comparison, all DRL baselines (DQN, PPO, CPO, RCPO, Lagrangian-PPO, FOCOPS, D3QN, DRL-RA) utilize the exact same MLP [256, 128, 64] architecture. This design decision nullifies network capacity discrepancies, ensuring performance variances reflect pure algorithmic merit rather than scale differences.

For hyperparameter fairness, we separate common settings from algorithm-specific settings. The network width, training horizon, number of seeds, evaluation workload, and state/action definitions are identical across learning-based methods. Method-specific parameters, such as PPO clipping and epochs, CPO trust-region radius, entropy coefficient, replay-buffer settings, and Lagrangian update rates, are selected from standard stable ranges recommended for each algorithm and then fixed before final testing. When default settings produce unstable learning, parameters are adjusted only within the same predefined stable ranges used for baseline tuning. All final comparisons use the same ten random seeds and the same evaluation traces, so the tuning budget remains comparable across methods.

After this tuning protocol is fixed, Table 6 reports the DRL-RA-specific values used in all final experiments, including the learning rates, replay setting, constraint budget, smoothness coefficient, and redundancy trigger.

The hyperparameters are selected from standard stable ranges used in DQN and constrained-RL implementations and then fixed for all random seeds. The learning rate 10^{-4} is used to avoid unstable Q-value updates in the mixed reliability/latency reward; the discount factor 0.99 reflects the long-horizon effect of queueing and resource reservation; the target-network update period and replay-buffer size follow common DQN stabilization practice. The smoothness value $\tau_{smooth} = 0.1$ is applied to normalized delay slack, so it corresponds to a transition band of roughly ten percent of the deadline.

Table 6: DRL-RA hyperparameters.

Hyperparameter	Value
Learning rate η_θ	10^{-4}
Lagrange multiplier rate η_λ	0.01
λ update frequency	Every 100 steps
Cost average window	100-step sliding average
Discount factor γ	0.99
Experience replay buffer size	10^5
Mini-batch size	64
Target network update frequency	500 steps
ε-greedy range	$[1.0, 0.01]$
ε decay rate	0.995
Cost budget d	0.05
Smoothness τ_{smooth}	0.1
Reward weights	$(0.3, 0.2, 0.3, 0.2)$

The cost budget $d = 0.05$ should be read as a long-term normalized smooth reliability-shortfall budget rather than as a hard per-task violation probability. Since the softplus cost is formed from the reliability gap after normalization, $d = 0.05$ means that the expected residual shortfall is constrained to a small margin around five percent of the unit-normalized cost scale. This is appropriate for SAGIN scheduling because transient visibility and channel fluctuations make a zero-violation target unrealistic, while the separate CVR metric still reports hard task-level violations.

Decision latency measurements are conducted on an Intel Xeon E5-2680 v4 CPU (2.4 GHz) with 64 GB RAM and an NVIDIA Tesla P100 GPU. Neural network inference uses PyTorch 1.12 with no model compression or quantization applied. The batch size is set to 1 (single-task decision scenario).

5.3 Evaluation Metrics

- **Task Completion Rate (TCR):** Percentage of tasks completed within deadlines.
- **Average Latency (AL):** Mean end-to-end latency (ms).
- **Energy Consumption (EC):** Average energy consumed per task (mJ).
- **System Reliability (SR):** Average $\mathcal{R}_{i,j}$ of completed tasks.
- **Resource Utilization (RU):** Mean utilization of edge servers.
- **Decision Latency (DL):** Time per decision (ms).
- **Constraint Violation Rate (CVR):** Proportion of tasks failing reliability thresholds.
- **Expected Cost $\mathbb{E}[\boldsymbol{c}]$:** Average reliability violation cost, verifying budget fulfillment.

5.4 Statistical Methodology

All outcomes are reported as means ± standard deviations across 10 random seeds. Statistical significance is evaluated with paired t-tests at $p < 0.05$, and confidence intervals are estimated using 1000 bootstrap resamples. In the main comparison table, an asterisk marker is added to DRL-RA results that are significantly better than the strongest comparable learning-based baseline under the same evaluation traces; confidence intervals are reported in the table note for the primary metrics.

6 Experimental Results and Analysis

This section reports the empirical evaluation. We first compare overall performance, then isolate the contribution of each component through ablation studies, and finally analyze sensitivity to reliability requirements, system size, workloads, and channel conditions.

6.1 Overall Performance Comparison

6.1.1 Experiment Design

This experiment systematically benchmarks DRL-RA against standard paradigms. Employing a strict control scheme guarantees fair comparisons within uniform environments. Specific settings:

We picked 12 benchmarks spanning three categories: (1) Classic methods including Random (random target selection), Greedy (nearest edge server), and Lyapunov optimization; (2) Unconstrained DRL (DQN, PPO); (3) Constrained RL (CPO, RCPO, Lagrangian-PPO, FOCOPS). Additionally, we benchmarked D3QN (Component A only) and RA-Opt (Component B only) to validate the integration.

Each learning-based method is trained with 10 random seeds for 1000 episodes and 1000 steps per episode. After training, each policy is evaluated over 10,000 time steps. Metrics include TCR, AL, EC, SR, RU, and DL. RA-Opt is solved online and is therefore reported as a non-real-time optimization reference for decision-quality assessment rather than as an inference-latency peer.

Results are reported in mean ± SD format. Significance is assessed via paired t-tests ($p < 0.05$), and 95% confidence intervals and Cohen's d are used to quantify effect size.

The overall comparison in Table 7 is used as the basis for the subsequent analysis because it reports not only task completion and latency, but also energy consumption, reliability, resource utilization, and decision latency under the same traces. This multi-metric view is necessary to distinguish a genuine reliability-aware gain from a policy that merely shifts cost to energy use or online computation time.

Table 7: Overall performance comparison (Mean ± SD across 10 seeds; * indicates $p < 0.05$ vs. the strongest comparable learning-based baseline).

Method	TCR (%)	AL (ms)	EC (mJ)	SR (%)	RU (%)	DL (ms)
Random	78.4 ± 2.3	1856 ± 156	246 ± 23	82.2 ± 1.8	45.2 ± 3.4	0.1 ± 0.0
Greedy	84.7 ± 1.9	1244 ± 98	198 ± 18	85.7 ± 1.5	62.9 ± 4.2	1.3 ± 0.2
Lyapunov	87.2 ± 1.6	988 ± 76	177 ± 15	87.9 ± 1.3	71.5 ± 3.8	15.7 ± 2.1
DQN	89.5 ± 1.4	892 ± 67	163 ± 12	86.3 ± 1.6	74.2 ± 3.5	2.2 ± 0.3
PPO	90.1 ± 1.3	867 ± 58	160 ± 11	87.2 ± 1.4	75.7 ± 3.2	2.9 ± 0.3
CPO	90.5 ± 1.2	845 ± 54	158 ± 10	90.8 ± 1.1	76.3 ± 3.0	4.2 ± 0.5
RCPO	91.2 ± 1.1	823 ± 51	155 ± 10	91.5 ± 1.0	77.1 ± 2.9	4.8 ± 0.5
Lagrangian-PPO	91.8 ± 1.0	801 ± 48	151 ± 9	92.3 ± 0.9	78.2 ± 2.7	3.9 ± 0.4
FOCOPS	91.5 ± 1.1	812 ± 50	153 ± 10	91.9 ± 1.0	77.8 ± 2.8	3.6 ± 0.4
D3QN (A Only)	89.5 ± 1.4	813 ± 52	152 ± 10	88.6 ± 1.5	77.3 ± 2.9	2.2 ± 0.3
RA-Opt (B Only)	89.8 ± 1.4	956 ± 68	171 ± 13	91.2 ± 1.2	73.1 ± 3.4	42.6 ± 5.3
DRL-RA (Full)	95.6 ± 0.8*	712 ± 42*	144 ± 8*	96.2 ± 0.7*	81.5 ± 2.4*	3.7 ± 0.4

Note: For DRL-RA, the 95% confidence intervals (estimated using 1000 bootstrap resamples over the seed-level results, $n = 10$ seeds) are TCR [95.0, 96.2]%, AL [682, 742] ms, EC [138.3, 149.7] mJ, SR [95.7, 96.7]%, RU [79.8, 83.2]%, and DL [3.4, 4.0] ms.

6.1.2 Result Analysis

Using the six metrics in Table 7, the analysis below first evaluates task completion and latency, then examines whether the gain is achieved without sacrificing energy, reliability, resource utilization, or deployment-time inference speed.

Table 7 illustrates performance comparisons across six core metrics.

DRL-RA achieves a TCR of 95.6 ± 0.8%, which is the highest value among the considered baselines. Compared with Lagrangian-PPO (91.8 ± 1.0%), the absolute improvement is 3.8 percentage points and is statistically significant ($p < 0.001$, Cohen's $d = 4.2$). The lower standard deviation of DRL-RA also indicates more stable behavior across random seeds. The comparison between constrained methods and unconstrained DQN/PPO supports the need for explicit reliability-cost modeling, while the additional gain of DRL-RA over Lagrangian-PPO indicates the benefit of combining the smooth reliability proxy with redundancy backup.

For average latency, DRL-RA obtains 712 ± 42 ms, which is 11.1% lower than Lagrangian-PPO (801 ± 48 ms). This result is consistent with the role of the dueling architecture in separating state-value and action-advantage estimation, which is useful when several offloading targets have similar action values. The smooth reliability term also avoids the abrupt policy changes that can occur with binary constraint indicators, allowing the policy to trade latency against reliability margins more gradually.

DRL-RA obtains the lowest energy consumption among the reported methods (144 ± 8 mJ). The reduction is consistent with fewer unnecessary long-haul transmissions and better load distribution across feasible edge, UAV, and satellite targets. Lyapunov optimization remains a useful queue-stability reference, but its 15.7 ± 2.1 ms online decision latency is higher than that of neural inference policies under the same hardware setting.

For system reliability, DRL-RA achieves 96.2 ± 0.7%, which is 7.6 percentage points higher than D3QN and 3.9 percentage points higher than Lagrangian-PPO. This result supports the contribution of RA-MOOF because the smooth proxy, Lagrangian update, and redundancy policy each target a different source of reliability degradation: marginal delay violations, long-term budget drift, and high-reliability task failure.

DRL-RA obtains the highest resource utilization (81.5 ± 2.4%) while maintaining the highest reliability. This indicates that the policy does not simply avoid loaded nodes; instead, it uses reliability-aware load distribution to exploit available resources while keeping the reliability cost within the budget.

Decision latency should be interpreted with the solver type in mind. DRL-RA requires 3.7 ± 0.4 ms for one neural-policy inference, while RA-Opt requires 42.6 ± 5.3 ms because it solves a mixed-integer optimization problem online. Thus, the RA-Opt latency value reflects online optimization cost, whereas deployment-time latency comparisons are primarily made among trained inference methods.

6.1.3 Constraint Budget Fulfillment Verification

To connect the surrogate feasibility discussion in Section 4.5 with empirical behavior, Table 8 compares the expected reliability cost $\mathbb{E}[c]$ with the budget d and also reports CVR for each constrained or unconstrained learning method.

To attest to CMDP constraint compliance, we profile expected costs $\mathbb{E}[c]$ vs. budget d across constrained RL schemes. Table 8 displays thorough constraint satisfaction stats.

The expected cost of DRL-RA is $\mathbb{E}[c] = 0.043 \pm 0.005$, which is below the budget $d = 0.05$. This verifies satisfaction of the surrogate expected-cost constraint used in Eq. (31). Unconstrained DQN and PPO exceed

the cost budget, while the constrained variants satisfy or nearly satisfy it; among them, DRL-RA has the lowest expected cost and the lowest CVR in Table 8.

Table 8: Constraint budget fulfillment verification (d = 0.05, Mean ± SD across 10 seeds).

Method	$\mathbb{E}[c]$	CVR (%)	Satisfies Constraint?
DQN	0.089 ± 0.012	14.3 ± 1.5	No
PPO	0.078 ± 0.010	11.8 ± 1.3	No
CPO	0.052 ± 0.008	5.2 ± 0.9	Borderline
RCPO	0.048 ± 0.007	4.8 ± 0.8	Yes
Lagrangian-PPO	0.046 ± 0.006	4.5 ± 0.7	Yes
FOCOPS	0.047 ± 0.007	4.6 ± 0.8	Yes
DRL-RA	0.043 ± 0.005	3.6 ± 0.8	Yes

6.1.4 Synergistic Effect between Components A and B

The comparison among D3QN, RA-Opt, and DRL-RA indicates complementary behavior. D3QN has low neural-inference latency (2.2 ± 0.3 ms) but lower reliability (88.6 ± 1.5%). RA-Opt provides an optimization-reference reliability of 91.2 ± 1.2%, while its 42.6 ± 5.3 ms decision time reflects the cost of online mixed-integer solving rather than neural-policy inference. DRL-RA preserves millisecond-level inference latency (3.7 ± 0.4 ms) while increasing reliability to 96.2 ± 0.7%, supporting the design choice of using D3QN for action selection and RA-MOOF for reliability-aware cost construction.

6.2 Comprehensive Ablation Experiments

6.2.1 Experiment Design

The ablation experiments evaluate the contribution of each DRL-RA module by removing or replacing one component at a time and reporting the resulting performance change. This single-factor design helps attribute the observed differences to the corresponding module while keeping the remaining training and evaluation settings unchanged.

The ablation table is organized to test this attribution logic directly: Table 9 groups removals by the D3QN decision module, the RA-MOOF reliability module, and their reward-level integration, so each performance drop can be linked to a specific design choice.

Ablation Configurations: We structured 12 setups across three tiers:

(1) **Component A (D3QN) Ablation:** Evaluates the Dueling architecture and Double Q-learning. The configurations remove Dueling while keeping Double Q-learning, remove Double Q-learning while keeping Dueling, or remove both to obtain vanilla DQN.

(2) **Component B (RA-MOOF) Ablation:** Evaluates the reliability-aware components. The configurations remove RA-MOOF entirely, replace the smooth proxy with a hard indicator, replace the adaptive Lagrangian multiplier with a fixed multiplier, or remove redundancy backup.

(3) **Integration Ablation:** Evaluates the interaction between the decision module and the reliability-aware objective by removing the reliability reward or the reliability penalty.

Metrics: Each configuration is evaluated over 10 independent seeds, and TCR, AL, and SR are reported. Performance changes are measured relative to the full DRL-RA configuration.

Table 9: Ablation study results (Mean ± SD, n = 10 seeds).

Configuration	TCR (%)	AL (ms)	SR (%)
Full DRL-RA	95.6 ± 0.8	712 ± 42	96.2 ± 0.7
Component A Ablations			
Remove Dueling Arch	93.9 ± 1.0	789 ± 51	94.5 ± 0.9
Remove Double Q-Learning	94.2 ± 0.9	757 ± 49	95.3 ± 0.8
Remove Both (Vanilla DQN)	91.3 ± 1.1	892 ± 67	86.3 ± 1.6
Component B Ablations			
Remove RA-MOOF (D3QN Only)	91.3 ± 1.1	813 ± 52	88.6 ± 1.5
Remove Smooth Proxy (Hard Ind)	92.8 ± 1.2	778 ± 55	93.4 ± 1.1
Remove Lagrangian (Fixed λ)	93.5 ± 1.0	756 ± 50	91.2 ± 1.2
Remove Redundancy Backup	94.1 ± 0.9	735 ± 47	94.9 ± 0.8
Integration Ablations			
Remove Reliability Reward	92.5 ± 1.1	767 ± 53	91.2 ± 1.1
Remove Reliability Penalty	93.4 ± 1.0	742 ± 49	92.8 ± 1.0

6.2.2 Component A (D3QN) Ablation Analysis

Removing the Dueling architecture reduces TCR from 95.6% to 93.9% (−1.7 percentage points), increases average latency from 712 to 789 ms (+10.8%), and reduces SR from 96.2% to 94.5% (−1.7 percentage points). This result is consistent with the role of Dueling networks in separating the state-value term $V(s)$ from the action-advantage term $A(s, a)$. In SAGIN offloading, several candidate nodes can have similar latency but different reliability or queue states; separating value and advantage helps distinguish such actions more stably.

Removing Double Q-learning decreases TCR from 95.6% to 94.2% (−1.4 percentage points), increases latency from 712 to 757 ms (+6.3%), and decreases SR from 96.2% to 95.3% (−0.9 percentage points). Double Q-learning decouples action selection from action evaluation and therefore reduces overestimation bias. In the offloading setting, reducing this bias helps avoid repeatedly selecting apparently high-value but congested nodes.

Removing both mechanisms, i.e., reverting to vanilla DQN, reduces TCR by 6.1 percentage points, increases latency by 25.3%, and reduces SR by 9.9 percentage points. The larger degradation compared with removing either mechanism alone suggests that Dueling and Double Q-learning provide complementary benefits: the former improves value decomposition, while the latter improves value-estimation stability.

6.2.3 Component B (RA-MOOF) Ablation Analysis

Replacing the smooth proxy with hard indicator functions reduces TCR from 95.6% to 92.8% (−2.8 percentage points) and SR from 96.2% to 93.4% (−2.8 percentage points). The standard deviation of SR also increases from 0.7% to 1.1%, indicating less stable training across random seeds. A hard indicator provides only sparse binary feedback, whereas the smooth proxy supplies continuous information about the degree of reliability deficit. This denser signal can reduce TD-target variance and yields smoother estimates for the Lagrangian cost term $\mathbb{E}[c]$.

Fixing the Lagrangian multiplier at $\lambda = 1$ reduces TCR from 95.6% to 93.5% (−2.1 percentage points) and SR from 96.2% to 91.2% (−5.0 percentage points). A fixed multiplier cannot adapt to different violation

levels: if it is too small, the policy under-penalizes reliability deficits; if it is too large, the policy may become overly conservative. The adaptive update adjusts the penalty according to the observed constraint cost, and in our runs λ moves from 1.0 to approximately the range [2.1, 2.5].

Removing redundancy reduces TCR from 95.6% to 94.1% (−1.5 percentage points) and SR from 96.2% to 94.9% (−1.3 percentage points). The effect is more visible for high-reliability tasks with $\rho^{min} \geq 0.95$: their completion rate decreases from 89.2% to 81.7% without redundancy. This supports the use of capacity-gated replicas for mission-critical tasks rather than for all tasks indiscriminately.

6.2.4 Integration Ablation Analysis

Removing the reliability reward reduces TCR from 95.6% to 92.5% (−3.1 percentage points) and SR from 96.2% to 91.2% (−5.0 percentage points). This indicates that the reliability reward helps the policy prefer nodes with higher reliability margins when several actions satisfy basic feasibility conditions.

Removing the reliability penalty reduces TCR from 95.6% to 93.4% (−2.2 percentage points) and SR from 96.2% to 92.8% (−3.4 percentage points). The immediate penalty and the Lagrangian multiplier act at different levels: the former shapes step-wise decisions, while the latter regulates the long-term expected constraint cost. Their combination therefore improves both local action selection and long-term constraint compliance.

6.2.5 Ablation Summary

The ablation results show five patterns: (1) the dueling and double-Q mechanisms jointly improve the discrete decision module; (2) the smooth proxy is important for stable reliability-cost learning; (3) adaptive Lagrangian updates help align reward optimization with the cost budget; (4) redundancy mainly benefits high-reliability tasks; and (5) reward and penalty weights control the tradeoff between performance and constraint compliance.

6.3 Sensitivity Analysis

The sensitivity analysis evaluates DRL-RA under different reliability requirements, system sizes, workload intensities, and channel conditions to examine its robustness within the simulated operating range.

6.3.1 Performance under Varied Reliability Needs

Design: We compare DRL-RA with Lagrangian-PPO and D3QN across five reliability thresholds: 0.80, 0.85, 0.90, 0.95, and 0.98. TCR and CVR are reported over 10 random seeds.

Motivation: Urban sensing applications have different reliability requirements. Environmental monitoring can tolerate moderate reliability thresholds, whereas public-safety tasks may require stricter thresholds such as $\rho^{min} \geq 0.98$. Evaluating multiple thresholds clarifies the reliability–performance tradeoff. Accordingly, Table 10 reports both TCR and CVR at each threshold, allowing the analysis to separate performance degradation caused by stricter reliability requirements from actual constraint-control behavior.

The two panels in Fig. 4 visualize the same threshold sweep from complementary perspectives: panel (a) highlights the TCR loss as reliability requirements tighten, whereas panel (b) shows whether this loss is accompanied by lower constraint violations.

Table 10: Performance under varied reliability requirements.

Method	Minimum Reliability Requirement				
	0.80	**0.85**	**0.90**	**0.95**	**0.98**
Task Completion Rate (%)					
D3QN	93.5 ± 1.1	91.3 ± 1.1	87.2 ± 1.3	79.7 ± 1.6	65.9 ± 2.1
Lagrangian-PPO	94.2 ± 1.0	91.8 ± 1.0	89.5 ± 1.1	85.3 ± 1.3	76.2 ± 1.7
DRL-RA	**96.8 ± 0.8**	**95.6 ± 0.8**	**93.1 ± 0.9**	**88.7 ± 1.1**	**82.5 ± 1.4**
Constraint Violation Rate (%)					
D3QN	14.3 ± 1.5	11.4 ± 1.3	10.8 ± 1.4	9.9 ± 1.5	8.6 ± 1.8
Lagrangian-PPO	8.2 ± 1.2	7.7 ± 1.1	6.4 ± 1.0	5.8 ± 1.1	4.9 ± 1.3
DRL-RA	**3.6 ± 0.8**	**3.2 ± 0.7**	**3.0 ± 0.7**	**2.8 ± 0.8**	**2.7 ± 0.9**

Note: Boldface indicates the best mean value at each reliability threshold (higher TCR and lower CVR are preferred). Boldface denotes a descriptive column-wise comparison; it does not by itself imply statistical significance.

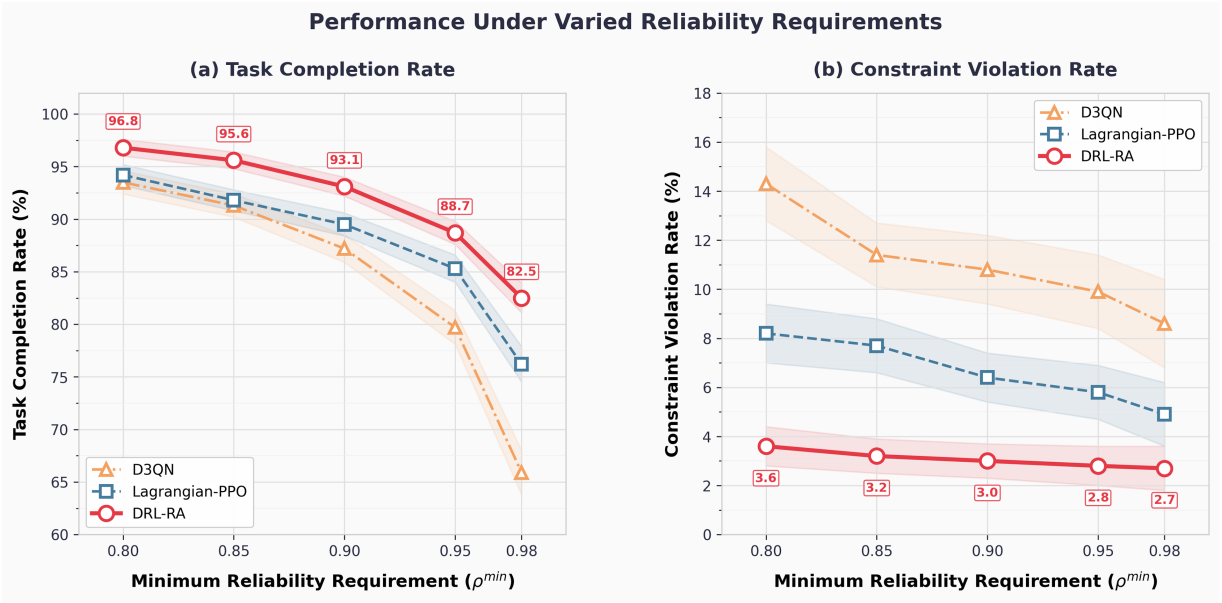

Figure 4: Performance under varied reliability requirements. (**a**) Task completion rate across different minimum reliability thresholds; (**b**) Constraint violation rate across different minimum reliability thresholds.

As the reliability requirement increases, all methods show lower TCR, but the degradation rates differ. D3QN decreases to 65.9% at $\rho^{min} = 0.98$, a 27.6-point drop from the threshold 0.80 case, whereas DRL-RA remains at 82.5%, corresponding to a 14.3-point drop. This indicates better robustness of DRL-RA under stricter reliability requirements in the simulated setting.

DRL-RA keeps CVR in the 3%–4% range and reaches 2.7 ± 0.9% at $\rho^{min} = 0.98$. D3QN shows higher violation rates across the same thresholds, while Lagrangian-PPO reduces violations but remains above DRL-RA in this simulation.

Using the TCR decrease per 0.1 increase in reliability threshold as a performance-loss ratio, DRL-RA loses 7.9 percentage points per 0.1 threshold increase, whereas D3QN loses 15.3 percentage points. This suggests that DRL-RA handles the reliability–performance tradeoff more effectively in this experiment.

At $\rho^{min} = 0.98$, DRL-RA completes 82.5% of tasks compared with 65.9% for D3QN. The improvement is consistent with the redundancy mechanism: the simulated system creates 1.8 replicas per high-reliability task on average at $\rho^{min} = 0.98$, compared with 1.1 replicas at $\rho^{min} = 0.80$.

6.3.2 Scalability across System Sizes

Design: We evaluate DRL-RA overhead and performance under four system sizes: (1) Small: 50 devices and 5 edge servers; (2) Medium: 100 devices and 10 edge servers; (3) Large: 200 devices and 15 edge servers; and (4) Ultra: 500 devices and 25 edge servers. Each setting is tested over 5 seeds, and TCR and DL are recorded.

The scalability curves in Fig. 5 show the expected decline in completion rate as the system expands, but they also allow the degradation of DRL-RA and D3QN to be compared under the same growth in devices and candidate offloading targets.

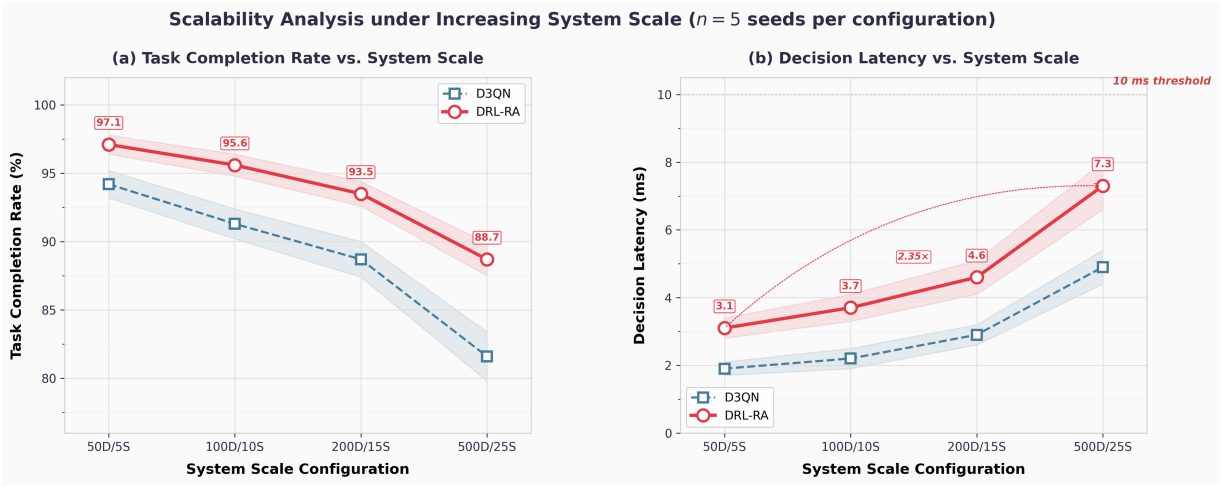

Figure 5: Scalability analysis (each group is configured with $n = 5$ seeds).

Larger systems reduce TCR for all methods because resource contention and the decision space both increase. When the number of devices grows from 50 to 500, DRL-RA decreases by 8.4 points (97.1% to 88.7%), whereas D3QN decreases by 12.6 points (94.2% to 81.6%). This indicates better scalability of the reliability-aware policy in the simulated system-size range.

Decision latency scales sub-linearly in this experiment. When the system grows from 50 to 500 devices, DRL-RA's latency increases from 3.1 to 7.3 ms (2.35×), while the action space increases from 6 to 26 actions (4.33×). This result is consistent with the batched matrix operations used in the neural-network forward pass. Even in the 500-device setting, the observed decision latency remains below 10 ms.

Training on the 500-device setting takes 3.2× longer than the medium-size baseline. The increase is mainly caused by environment interaction and experience collection rather than by the policy update itself, suggesting that parallelized data collection could further reduce training time.

6.3.3 Robustness under Workloads and Channel Noise

Design: We evaluate varying workloads and channel SNR conditions. The arrival rates are Low (5 tasks/s/device), Medium (12.5 tasks/s/device), and High (25 tasks/s/device). The SNR levels are Low (5 dB), Medium (15 dB), and High (25 dB). Each setting is evaluated over 10 seeds.

The robustness plots in Fig. 6 connect two different stress factors to the same performance metrics: workload growth mainly increases queueing pressure, whereas lower SNR directly weakens link reliability and increases transmission time.

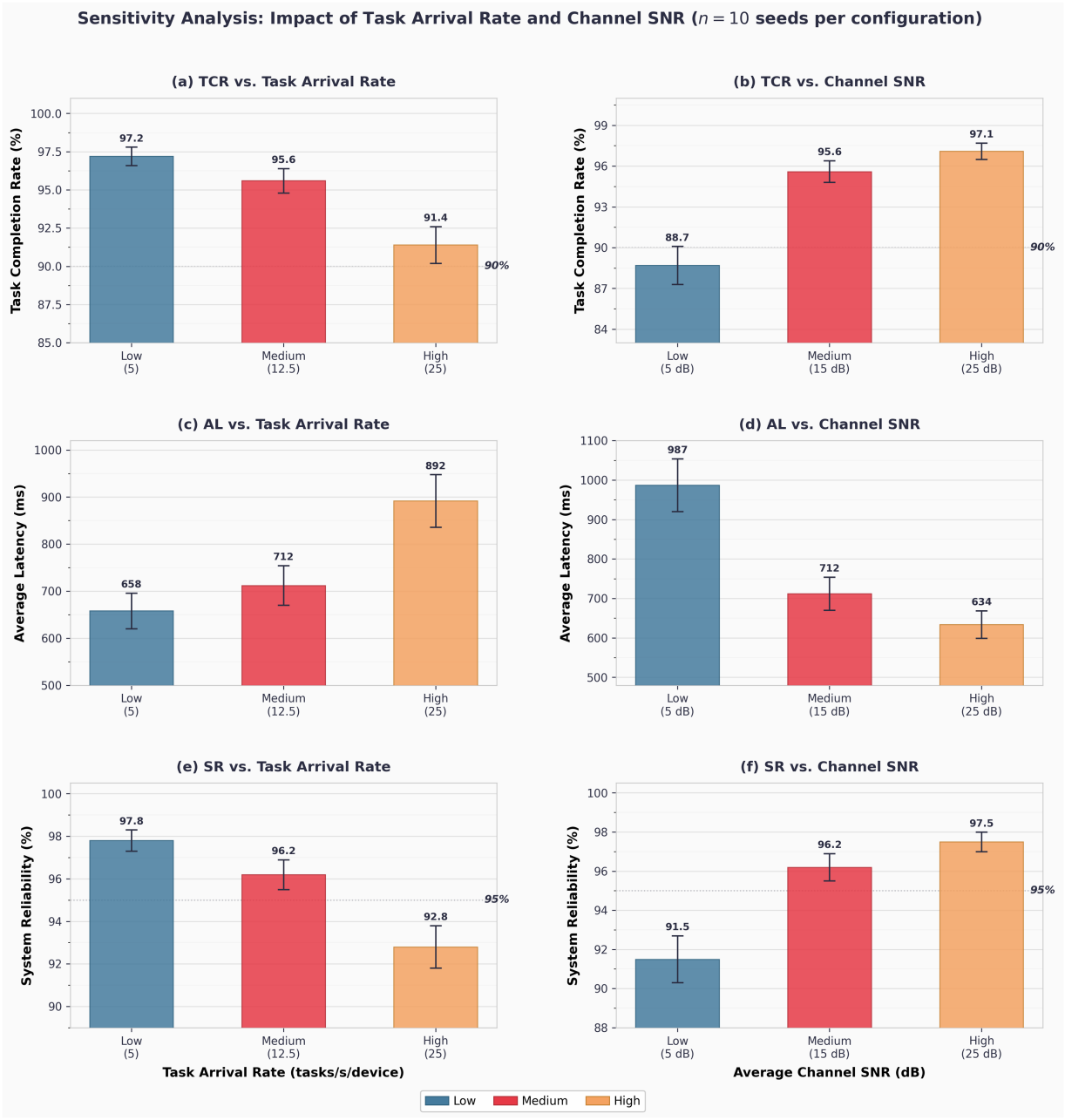

Figure 6: Sensitivity to task arrival rate and channel SNR.

Under high arrival rates (25 tasks/s/device), queues become longer and task drop risk increases. DRL-RA obtains a 91.4% TCR, which is 5.8 percentage points lower than the low-load case. The result is consistent with load balancing across feasible nodes. Average latency increases by 35.6%, while SR decreases by 4.4 percentage points, showing that reliability is less sensitive than latency to workload growth in this simulation.

Low SNR (5 dB) reduces link reliability $\psi_{i,j}$ and increases transmission time. DRL-RA obtains an 88.7% TCR in this setting, which is 8.4 percentage points lower than the high-SNR case. Latency increases by 55.7%, and the policy compensates by selecting local execution or alternative links when their estimated reliability is higher.

Using a robustness index defined as fluctuation range divided by baseline performance, DRL-RA obtains 5.9% under arrival-rate variation and 8.8% under SNR variation. The larger fluctuation under SNR changes is consistent with the direct effect of channel quality on link reliability and transmission time.

7 Conclusion

This paper presented DRL-RA, a reliability-aware and low-latency task offloading and resource allocation framework for SAGIN. The method combines D3QN-based discrete action selection with RA-MOOF-based reliability cost construction, smooth surrogate constraint handling, and capacity-gated redundancy for high-reliability tasks. The system model explicitly captures heterogeneous communication links, hierarchical computing nodes, satellite visibility windows, UAV availability, and composite reliability under stated conditional-independence assumptions. Simulation results across multiple random seeds show that DRL-RA improves task completion rate, latency, and system reliability over the implemented baselines, while the constraint-budget table verifies satisfaction of the surrogate expected-cost constraint. The problem formulation specifies the observation and update assumptions under which satellite visibility windows, UAV availability, queue states, and link measurements are used for repeated short-horizon decisions. Future work can further extend this formulation to explicitly model correlated large-scale disruptions.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Fei Bu and Zheng Wang; methodology, Zheng Wang and Yong Pan; software, Zhaomin Wu and Yuchen Liang; validation, Fei Bu and Zhaomin Wu; formal analysis, Zheng Wang; investigation, Fei Bu, Yong Pan and Zhongshan Zhu; resources, Zhongshan Zhu; data curation, Yuchen Liang and Zhaomin Wu; writing—original draft preparation, Fei Bu and Zheng Wang; writing—review and editing, Zheng Wang and Tengfei Tu; visualization, Zhaomin Wu and Yuchen Liang; supervision, Zheng Wang and Tengfei Tu; project administration, Fei Bu; funding acquisition, Fei Bu and Zheng Wang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the Corresponding Author, Zheng Wang, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhang Z, Xiao Y, Ma Z, Xiao M, Ding Z, Lei X, et al. 6G wireless networks: vision, requirements, architecture, and key technologies. IEEE Veh Technol Mag. 2019;14(3):28–41. doi:10.1109/MVT.2019.2921208.
2. Liu J, Shi Y, Fadlullah ZM, Kato N. Space-air-ground integrated network: a survey. IEEE Commun Surv Tut. 2018;20(4):2714–41. doi:10.1109/comst.2018.2841996.
3. Cai Y, Cheng P, Chen Z, Xiang W, Vucetic B, Li Y. Graphic deep reinforcement learning for dynamic resource allocation in space-air-ground integrated networks. IEEE J Sel Areas Commun. 2025;43(1):334–49. doi:10.1109/jsac.2024.3460086.
4. Li H, Yu J, Cao L, Zhang Q, Song Z, Hou S. Multi-agent reinforcement learning based computation offloading and resource allocation for LEO satellite edge computing networks. Comput Commun. 2024;222(4):268–76. doi:10.2139/ssrn.4611047.
5. Huang C, Chen G, Xiao P, Xiao Y, Han Z, Chambers JA. Joint offloading and resource allocation for hybrid cloud and edge computing in SAGINs: a decision assisted hybrid action space deep reinforcement learning approach. IEEE J Sel Areas Commun. 2024;42(5):1029–43. doi:10.1109/JSAC.2024.3365899.
6. Liu L, Mao W, Li W, Duan J, Liu G, Guo B. Edge computing offloading strategy for space-air-ground integrated network based on game theory. Comput Netw. 2024;243(10):110331. doi:10.1016/j.comnet.2024.110331.
7. Wachi A, Shen X, Sui Y. A survey of constraint formulations in safe reinforcement learning. In: Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI); 2024 Aug 3–9; Jeju, Republic of Korea. p. 8262–71.
8. Mao Y, Zhang J, Letaief KB. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J Sel Areas Commun. 2016;34(12):3590–3605. doi:10.1109/jsac.2016.2611964.
9. Chen X. Decentralized computation offloading game for mobile cloud computing. IEEE Trans Parallel Distrib Syst. 2015;26(4):974–83. doi:10.1109/tpds.2014.2316834.
10. Zhang J, Hu X, Ning Z, Ngai ECH, Zhou L, Wei J, et al. Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks. IEEE Internet Things J. 2018;5(4):2633–45. doi:10.1109/jiot.2017.2786343.
11. Geng L, Zhao H, Wang J, Kaushik A, Yuan SWK, Feng W. Deep reinforcement learning-based distributed computation offloading in vehicular edge computing networks. IEEE Internet Things J. 2023;10(14):12416–33. doi:10.1109/jiot.2023.3247013.
12. Xiao Y, Ye Z, Wu M, Li H, Xiao M, Alouini MS, et al. Space-air-ground integrated wireless networks for 6G: basics, key technologies, and future trends. IEEE J Sel Areas Commun. 2024;42(12):3327–54.
13. Zhang J, Yang X, Chen X, Chen X, Yi X, Khalil I, et al. Energy-efficient UAV deployment and computation offloading in space-air-ground integrated networks. IEEE Trans Veh Technol. 2026;75(2):3081–98. doi:10.1109/tvt.2025.3601197.
14. Zhang P, Li Y, Kumar N, Chen N, Hsu CH, Barnawi A. Distributed deep reinforcement learning assisted resource allocation algorithm for space-air-ground integrated networks. IEEE Trans Netw Serv Manag. 2023;20(3):3348–58. doi:10.1109/tnsm.2022.3232414.
15. Liu Y, Jiang L, Qi Q, Xie K, Xie S. Online computation offloading for collaborative space/aerial-aided edge computing toward 6G system. IEEE Trans Veh Technol. 2024;73(2):2495–505. doi:10.1109/tvt.2023.3312676.
16. Zhang X, Liu J, Zhang R, Huang Y, Tong J, Xin N, et al. Energy-efficient computation peer offloading in satellite edge computing networks. IEEE Trans Mob Comput. 2024;23(4):3077–91. doi:10.1109/tmc.2023.3269801.
17. Bhattacherjee D, Aqeel W, Bozkurt IN, Aguirre A, Chandrasekaran B, Godfrey PB, et al. Gearing up for the 21st century space race. In: Proceedings of the 17th ACM Workshop on Hot Topics in Networks. New York, NY, USA: ACM; 2018. p. 113–9.
18. Mozaffari M, Saad W, Bennis M, Nam YH, Debbah M. A tutorial on UAVs for wireless networks: applications, challenges, and open problems. IEEE Commun Surv Tut. 2019;21(3):2334–60.
19. Li M, Cheng N, Gao J, Wang Y, Zhao L, Shen X. Energy-efficient UAV-assisted mobile edge computing: resource allocation and trajectory optimization. IEEE Trans Veh Technol. 2020;69(3):3424–38.
20. Yan M, Xiong R, Wang Y, Li C. Edge computing task offloading optimization for a UAV-assisted Internet of Vehicles via deep reinforcement learning. IEEE Trans Veh Technol. 2024;73(4):5647–58. doi:10.1109/tvt.2023.3331363.

21. Mao Y, You C, Zhang J, Huang K, Letaief KB. A survey on mobile edge computing: the communication perspective. IEEE Commun Surv Tut. 2017;19(4):2322–58. doi:10.1109/comst.2017.2745201.
22. Zhang Y, Hu J, Min G. Digital twin-driven intelligent task offloading for collaborative mobile edge computing. IEEE J Sel Areas Commun. 2023;41(10):3034–45. doi:10.1109/jsac.2023.3310058.
23. Chen X, Jiao L, Li W, Fu X. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Netw. 2016;24(5):2795–808. doi:10.1109/tnet.2015.2487344.
24. Xie M, Ye J, Zhang G, Ni X. Deep reinforcement learning-based computation offloading and distributed edge service caching for mobile edge computing. Comput Netw. 2024;250(9):110564. doi:10.1016/j.comnet.2024.110564.
25. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature. 2015;518(7540):529–33. doi:10.1038/nature14236.
26. Chen X, Zhang H, Wu C, Mao S, Ji Y, Bennis M. Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. IEEE Internet Things J. 2018;6(3):4005–18. doi:10.1109/jiot.2018.2876279.
27. Zhu X, Zhang T, Zhang J, Zhao B, Zhang S, Wu C. Deep reinforcement learning-based edge computing offloading algorithm for software-defined IoT. Comput Netw. 2023;235:110006. doi:10.21203/rs.3.rs-2150294/v1.
28. Ratnabala L, Peter R, Fedoseev A, Tsetserukou D. Hippo-mat: decentralized task allocation using GraphSAGE and multi-agent deep reinforcement learning. arXiv:2503.07662. 2025.
29. Meng K, Zhang S, Li R, Meng X, Deng Y, Wang C, et al. Multi-agent conditional diffusion model with mean field communication as wireless resource allocation planner. arXiv:2510.22969. 2025.
30. Jia K, Xia H, Zhang R, Sun Y, Wang K. Multi-agent DRL for edge computing: a real-time proportional compute offloading. Comput Netw. 2024;252:110665.
31. Shao Z, Yang H, Xiao L, Su W, Chen Y, Xiong Z. Deep reinforcement learning-based resource management for UAV-assisted mobile edge computing against jamming. IEEE Trans Mob Comput. 2024;23(12):13358–70. doi:10.1109/globecom54140.2023.10437090.
32. Achiam J, Held D, Tamar A, Abbeel P. Constrained policy optimization. In: ICML'17: Proceedings of the 34th International Conference on Machine Learning; 2017 Aug 6–11; Sydney, NSW, Australia. p. 22–31.
33. Tessler C, Mankowitz DJ, Mannor S. Reward constrained policy optimization. arXiv:1805.11074. 2018.
34. Zhang Y, Vuong Q, Ross K. First order constrained optimization in policy space. Adv Neural Inf Process Syst. 2020;33:15338–49. doi:10.52202/075280-1703.
35. Cao K, Chen M, Karnouskos S, Hu S. Reliability-aware personalized deployment of approximate computation IoT applications in serverless mobile edge computing. IEEE Trans Comput Aided Des Integr Circ Syst. 2025;44(2):430–43. doi:10.1109/tcad.2024.3437344.
36. Li Z, Yu H, Fan G, Zhang J, Xu J. Energy-efficient reliability-aware offloading for delay-sensitive tasks in collaborative edge computing. Concurr Comput: Pract Exp. 2024;36(13):e8083. doi:10.1002/cpe.8083.
37. Tang J, Nie J, Zhang Y, Xiong Z, Jiang W, Guizani M. Multi-UAV-assisted federated learning for energy-aware distributed edge training. IEEE Trans Netw Serv Manag. 2024;21(1):280–93. doi:10.1109/tnsm.2023.3298220.
38. Qu Y, Zhang T, Feng Y, Xu T, Guo Z. Computation offloading and resource allocation for E2E tasks in satellite edge computing networks. Space: Sci Technol. 2024;4:0144.
39. Xue Z, Liu C, Liao C, Han G, Sheng Z. Joint service caching and computation offloading scheme based on deep reinforcement learning in vehicular edge computing systems. IEEE Trans Veh Technol. 2023;72(5):6709–22. doi:10.1109/tvt.2023.3234336.
40. Wang Y, Sheng M, Wang X, Wang L, Li J. Mobile-edge computing: partial computation offloading using dynamic voltage scaling. IEEE Trans Commun. 2016;64(10):4268–82.
41. 3GPP. Study on channel model for frequencies from 0.5 to 100 GHz. 3GPP TR 38.901, Version 19.2.0; 2026 [cited 2026 May 10]. Available from: https://www.etsi.org/deliver/etsi_tr/138900_138999/138901/19.02.00_60/tr_138901v190200p.pdf.
42. ITU-R. Recommendation ITU-R P.618-14: propagation data and prediction methods required for the design of Earth-space telecommunication systems. International Telecommunication Union; 2023 [cited 2026 May 10]. Available from: https://www.itu.int/rec/R-REC-P.618-14-202308-I/en.

43. Atzori L, Iera A, Morabito G. The internet of things: a survey. Comput Netw. 2010;54(15):2787–805. doi:10.1016/j.comnet.2010.05.010.
44. Khan LU, Yaqoob I, Tran NH, Kazmi SMA, Dang TN, Hong CS. Edge-computing-enabled smart cities: a comprehensive survey. IEEE Internet Things J. 2020;7(10):10200–232.
45. Abbas N, Zhang Y, Taherkordi A, Skeie T. Mobile edge computing: a survey. IEEE Internet Things J. 2017;5(1):450–65. doi:10.1109/jiot.2017.2750180.