Cross-Layer Task Scheduling for NOMA-Assisted Satellite Edge Computing

Ruizhi Wang, Xiaolong Xu, *Senior Member, IEEE,* Guangming Cui, Muhammad Bilal, *Senior Member, IEEE,* and Fei Dai

***Abstract*—Ubiquitous, low-latency intelligence at the network edge is central to large-scale Internet of Things (IoT) deployments, yet effectively coordinating communication, computing, and backhaul operations across heterogeneous layers remains challenging. This paper presents a unified cross-layer framework for terrestrial–satellite edge computing IoT systems that integrates Non-Orthogonal Multiple Access (NOMA)-based terrestrial access with local, satellite, cloud execution. The framework jointly determines path selection and partial offloading to minimize a latency–energy objective using accurate end-to-end models. Within this framework, two complementary scheduling algorithms are developed. The Centralized Optimal Cross-Layer Scheduler (COCS) formulates the joint scheduling problem as a mixed-integer nonlinear program (MINLP) with logarithmic and bilinear terms. It solves it exactly using an exact solver, serving as a performance benchmark. The Decentralized Game-Theoretic Scheduler (DGTS) models user decisions as an ordinal potential game (OPG) and achieves distributed convergence via best-response dynamics (BRD), enabling scalability and adaptability to large networks. Extensive simulations demonstrate that COCS achieves the global optimum while DGTS attains near-optimal performance with much lower complexity. Together, validate the effectiveness of the proposed cross-layer framework and highlight the importance of coordinated communication–computation–backhaul design for terrestrial–satellite integrated edge computing.**

***Index Terms*—Internet of Things, Game-Theoretic Scheduling, NOMA, hybrid ground-satellite networks, LEO satellites**

I. Introduction

With the continuous proliferation of mobile Internet, large-scale Internet of Things (IoT), and data-intensive intelligent applications, user demands for ubiquitous connectivity and real-time computation have grown rapidly [1]–[3]. Applications such as augmented reality (AR), virtual reality (VR), real-time video analytics, and autonomous driving require high throughput and ultra-low latency that traditional mobile devices and centralized cloud computing architectures can no longer guarantee [4]. These workloads appear as heterogeneous sensing/actuation loops with massive machine-type communications (mMTC)–scale concurrency and ultra-reliable low-latency communications (URLLC)–style deadlines, which pushes computation from the cloud to the device–edge–cloud continuum. Multi-access Edge Computing (MEC) has emerged as a promising paradigm that brings computing and storage resources closer to users, effectively reducing transmission delay and backbone network congestion. Nevertheless, conventional MEC infrastructures mainly rely on terrestrial base stations (BSs) and fiber backhaul, which are limited by geographic coverage, high deployment costs, and vulnerability to failures in extreme or remote environments [5]. These limitations highlight the need for new architectures capable of balancing end-to-end delay, instantaneous throughput, and energy efficiency through the migration of computation toward the network edge.

A promising solution to this limitation is the integration of satellite networks with terrestrial edge computing systems. Unlike simply deploying more terrestrial BSs or fiber links, incorporating satellite resources introduces a new geographically distributed reconfigurability across ground and space. This enables adaptive coverage, backhaul routing, and computing placement—providing greater elasticity and resilience than conventional MEC alone. Low Earth Orbit (LEO) satellites, with their global coverage and low latency, offer a unique opportunity to enhance the flexibility and scalability of edge computing systems [6]. In practice, the propagation delay of LEO links allows the satellite–edge–cloud tri-layer to operate within a shared latency budget, enabling flexible task placement policies. By offloading computational tasks to satellites, IoT endpoints in coverage-sparse or infrastructure-constrained areas can tap satellite or cloud compute while preserving low end-to-end latency over the non-terrestrial link, enabling wide-area IoT beyond terrestrial footprints. Moreover, by integrating satellite-based MEC with traditional terrestrial edge computing systems, we can create a hybrid network that leverages the strengths of both ground and space-based resources. For massive IoT, this hybridization turns connectivity, backhaul, and compute placement into a system knob to meet device power budgets and deadline reliability.

Despite the promising potential of hybrid ground–satellite systems, persistent challenges in task offloading, resource allocation, and communication management still constrain their practical benefits. In IoT deployments, Offloading decisions are limited by heterogeneous device capabilities and the constrained computing and caching resources of satellite edge nodes, making it difficult to maintain low latency under heavy demand [7], [8]. Resource allocation further complicates the problem, as bandwidth, power, and CPU cycles must be jointly optimized; decoupled or static allocation easily leads to bottlenecks that negate expected performance gains [9]. On the communication side, while Non-Orthogonal Multiple Ac-

(Corresponding author: Guangming Cui.)

Ruizhi Wang, Xiaolong Xu, and Guangming Cui are with the School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China. (e-mail: rzwdm3ky@163.com, xlxu@ieee.org, gcui@nuist.edu.cn).

Muhammad Bilal is with the School of Computing and Communications, Lancaster University, LA1 4WA, Lancaster, U.K. (e-mail: m.bilal@ieee.org).

Fei Dai is with the College of Big Data and Intelligent Engineering, Southwest Forestry University, Kunming 650224, China. (e-mail: daifei@swfu.edu.cn).

cess (NOMA) improves spectral efficiency, residual inter-user interference and imperfect successive interference cancellation (SIC) under realistic conditions reduce achievable throughput and energy efficiency [10]–[12], which is especially critical for mission-critical IoT, where rate degradation directly translates to deadline violation and elevated device energy. More importantly, NOMA user association and power control directly affect achievable data rates, which in turn constrain task partitioning deadlines. Adjusting task ratios then changes network load and interference levels, forming a closed-loop dependency among communication, computation, and caching that ultimately governs IoT service reliability and battery lifetime. Any single-layer optimization thus risks suboptimal or unstable outcomes. These interdependencies make cross-layer coordination among computation, communication, and resource management a key obstacle to fully realizing the benefits of ground–satellite integration.

Although extensive research has addressed these challenges—ranging from offloading optimization [7], [8] and resource scheduling [6], [9] to NOMA-based transmission design [9], [11], [12]—most existing works still treat these layers independently or assume idealized conditions such as fixed channel states. Such simplifications fail to capture the coupling effects among layers, and centralized optimization models like mixed-integer nonlinear program (MINLP) remain computationally intractable for large-scale systems [13]. What is needed, therefore, is not merely a faster solver but a structured modeling approach that exposes problem decomposability and supports scalable, practically implementable decision mechanisms.

To bridge this gap, this paper proposes a unified hybrid ground–satellite edge computing framework for IoT systems that enables joint optimization across communication and computation domains. Building on this architecture, we design two complementary algorithms—a centralized optimization scheme called the Centralized Optimal Cross-Layer Scheduler (COCS), which serves as the global performance benchmark, and a decentralized scheduling mechanism, named the Decentralized Game-Theoretic Scheduler (DGTS), which supports dense IoT and latency-sensitive tasks, adaptive decision-making under realistic constraints. The overall framework jointly considers latency and energy consumption, laying the foundation for the system modeling and optimization formulations presented in the following sections.

The primary contributions of this paper are as follows:

1) We propose a hybrid ground–satellite edge computing architecture built upon NOMA-based uplink transmission and multi-layer task offloading, where tasks can be flexibly offloaded to local devices, edge servers, or cloud centers under a unified cross-layer latency–energy objective tailored to IoT workloads.
2) We develop two complementary scheduling algorithms: The first one, named COCS, initially formulates the joint communication–computation–backhaul optimization as a MINLP and implements it as a nonconvex mixed-integer quadratically constrained program (MIQCP) for exact global solution via a commercial solver; and the second one named DGTS, models user interactions as an ordinal potential game (OPG) and achieves distributed convergence through best-response dynamics (BRD), providing a low-complexity, scalable solution for dense IoT deployments.
3) Extensive simulations demonstrate that COCS serves as a tight global benchmark, while DGTS achieves near-optimal performance with significantly lower complexity, validating the effectiveness and practicality of the proposed cross-layer framework from an IoT system perspective.

This paper is structured as follows: Section IV provides an in-depth discussion of the system model and the problem formulation. In Section V, we introduce the game-theoretic approach used for task offloading, including the algorithmic details and convergence analysis. Section VI presents the experimental setup and performance evaluation, comparing the proposed approach with baseline algorithms in both small-scale and large-scale scenarios. Finally, Section VII concludes the paper and discusses potential future directions.

II. Related Work

Recent systems work has revealed the practical constraints of LEO satellite connectivity that shape end-to-end performance. Xie et al. [14] model a satellite–terrestrial integrated network with queueing-aware state dynamics and design a delay-driven DRL offloading policy together with a multi-level feedback compute allocator, showing consistent energy savings while explicitly capturing coupling between radio rates, queueing delay, and compute allocation. Yao et al. [6] build a satellite–ground cooperation platform for AI inference that adapts model footprints per satellite and schedules inference across moving satellites and ground edges, reducing variance in end-to-end completion time and mitigating mobility-induced transmission failures. On the device and edge side of the stack, Xing et al. [15] empirically dissect the feasibility of in-orbit computing using commercial off-the-shelf components and quantify the thermal, radiation, and power constraints that bound compute offload to satellites. Several data-plane studies focus on LEO support for massive IoT: Shenoy et al. [16] design a constellation-aware medium access scheme that adapts to satellite visibility dynamics, while Ren et al. [17] build a smartphone-to-LEO IoT pipeline and identify link- and protocol-level bottlenecks introduced by mobility and narrowband uplinks. Together, these results motivate hybrid ground–satellite architectures where task placement and radio scheduling must react to satellite mobility, PoP locality, and inter-satellite link (ISL) usage rather than static link assumptions.

Surveys on orbital edge computing and joint communication–computing in space–air–ground networks systematize the design space spanning resource fragmentation across moving nodes, intermittent links, and heterogeneous compute [18], [19]. Building on this view, Singh et al. [20] advocate spectrum- and topology-aware scheduling to scale satellite IoT backhaul under tight bandwidth budgets. These works emphasize that practical gains hinge on cross-layer control that couples radio access, backhaul routing, and

computing placement—an assumption our modeling and algorithms explicitly adopt via joint path selection and offloading ratio optimization.

Within terrestrial and integrated networks, a large body of work studies NOMA-assisted MEC under reliability or latency constraints. Dong et al. [21] jointly optimize task offloading and power allocation for reliability-aware services, capturing queueing-latency violation and decoding error probabilities, and show energy reductions under realistic traffic. Kim et al. [22] investigate random access and shared-uplink resource allocation with NOMA under load bursts, exposing the sensitivity of throughput and delay to contention windows and power-domain groupings. Beyond baseline formulations, Huang et al. [23] combine power allocation with task replication to improve deadline reliability in vehicular edge settings where fast fading and mobility induce frequent SIC ordering changes. These studies consistently indicate that imperfect SIC and residual interference propagate to deadline feasibility and energy budgets, reinforcing the need to co-design association, power, and task partitioning. Our framework captures these couplings explicitly and further integrates satellite paths and backhaul latencies.

Meeting strict end-to-end deadlines over multi-hop wireless networks remains challenging due to interference coupling and route activation combinatorics. Tsanikidis et al. [13] develop near-optimal algorithms with provable constant-factor guarantees for stochastic deadline traffic in multi-hop wireless networks, clarifying when regularized schedules suffice. In [24], Jones et al. study slicing and scheduling with service guarantees, characterizing feasibility regions and showing that almost-regular schedules are optimal under delay-deficit metrics. On the service-provisioning side, Chen et al. [25] propose an online controller that adapts to non-stationary traffic and channels via transfer learning, improving quality of service (QoS) for non-terrestrial services under model drift. These lines of work motivate our design choices: we avoid centralized heavy-weight solvers in the critical path by using a decentralized best-response mechanism that converges to stable operating points, while still approaching the centralized MINLP benchmark.

Compared with the above, this paper fills a gap between measurement-driven satellite connectivity insights and NOMA-MEC formulations that often abstract away space-layer latencies or treat communication and computation separately.

III. Motivation Example

We ground the discussion in a representative IoT scenario to illustrate the interplay between heterogeneous access conditions, multi-tier computation, and task offloading decisions in a hybrid ground–satellite system. For ease of reference, Fig. 1 outlines the system layers involved.

To keep the modeling concrete yet simple, consider a small scene with two terrestrial BSs (B_1 and B_2), one LEO satellite (S_1), a remote cloud center (C), and five representative user equipments (UEs), which are interrupted here as IoT endpoints under heterogeneous conditions. BSs are used only to relay

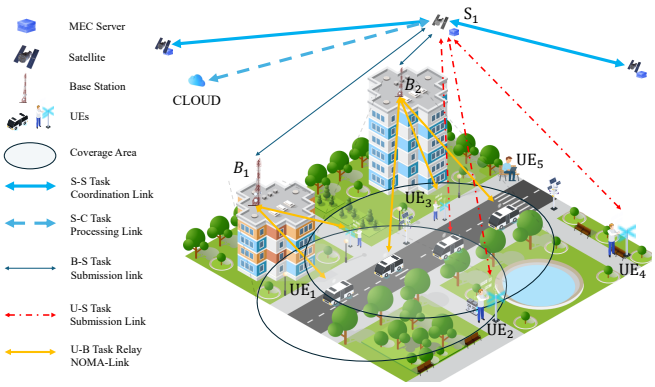

Fig. 1. Hybrid ground–satellite edge computing architecture integrating device, edge, and cloud layers for IoT workloads.

uplink traffic; execution may occur on the device, at the satellite edge, or in the cloud. We refer to five path types for end-to-end execution: local on-device (U); direct to satellite with satellite-edge execution (US); via a BS to the satellite with satellite-edge execution (UBS); direct to satellite and then to the cloud (USC); and via a BS to the satellite and then to the cloud (UBSC). The UE→BS hop uses power-domain NOMA with SIC, so users sharing the same BS can be coupled on that hop; in contrast, the UE→Satellite, BS→Satellite, and Satellite→Cloud segments are treated as orthogonal. These five paths cover canonical IoT execution patterns (local-only, device–edge, and device–edge–cloud) under mixed connectivity and compute availability.

UE_1 is a latency-sensitive near-cell user attached to B_1. Ground access is strong and stable, but S_1 is already ingesting multiple offloaded tasks (notably from UE_3 and occasional bursts from UE_4). To avoid waiting in the satellite queue and to tap abundant compute, UE_1 sends the offloaded portion via B_1 and S_1 to the cloud and keeps a small fraction local in parallel. In short, UE_1 chooses UBSC: relay where the uplink is strongest, then execute in the cloud to sidestep satellite-side contention.

UE_2 sits at the cell edge of B_1 and shares the NOMA uplink with other users. Its effective UE→BS performance is sensitive to the SIC order and concurrent load at B_1. To decouple from this shared bottleneck, UE_2 bypasses B_1 and transmits directly to S_1. Because S_1 is concurrently handling tasks from other UEs, and UE_2's job is sizeable enough that queuing at the satellite would be undesirable, the task is forwarded on to the cloud for execution. Thus, UE_2 chooses USC: direct satellite access to avoid ground contention, with computation offloaded to the cloud to absorb the workload spike at the satellite.

UE_3 connects to B_2 and generates a moderate-size analytics task. In this snapshot, B_2 is lightly loaded and provides a predictable uplink, while S_1 has capacity for medium tasks because heavy flows (e.g., UE_1 and UE_2) are pushed onward to the cloud. Finishing at the satellite avoids additional backhaul and keeps end-to-end latency low without consuming cloud resources. Consequently, UE_3 chooses UBS: relay through B_2 and execute at the satellite edge.

UE_4 operates without terrestrial coverage and can only transmit directly to the satellite. Its task is an incremental

imaging workload consisting of periodically captured small image tiles. For this kind of workload, satellite-edge execution is appropriate and avoids involving the cloud entirely. When a large batch of raw images arrives, however, S_1 may forward the bulk of the data to the cloud while retaining only preprocessing locally. In the present illustration, we focus on the incremental imaging case, so UE_4 chooses US: direct satellite access with execution at the satellite edge, while acknowledging that a surge would naturally shift this to USC for the heavier portion.

UE_5 carries a tiny and privacy-sensitive job. For such cases, shipping data provides little benefit and introduces unnecessary transmission and backhaul exposure. The device therefore computes entirely on its own, avoiding network resources and meeting its needs with minimal overhead. UE_5 chooses U: all-local execution.

This self-contained vignette maps each UE to a single path and explains the immediate reason for that choice in terms of coverage, shared uplink contention at the BS, instantaneous load at the satellite, task size, and IoT-specific constraints such as battery, duty-cycling, and privacy. In short: $\mathrm{UE}_1 \rightarrow$ UBSC (strong ground access; satellite busy; cloud absorbs compute), $\mathrm{UE}_2 \rightarrow$ USC, $\mathrm{UE}_3 \rightarrow$ UBS (lightly loaded B_2; satellite has room for medium tasks), $\mathrm{UE}_4 \rightarrow$ US (no BS coverage; satellite-edge execution for incremental imaging, with cloud as a fallback for surges), and $\mathrm{UE}_5 \rightarrow$ U (tiny or sensitive workload; stay local).

IV. System Model and Problem Formulation

This section presents the overall system model and problem formulation [26]. We first describe the network structure and notation, followed by channel and rate models. Then, we detail the execution modes and derive the corresponding latency and energy expressions [27]–[29], leading to an optimization formulation that jointly minimizes delay and energy consumption.

A. IoT System Architecture and Notation

We investigate a hybrid ground–satellite edge computing architecture tailored for IoT workloads, as illustrated in Fig. 1. The system consists of a set of user equipments (UEs) $\mathcal{U} = \{1, \ldots, U\}$, a set of terrestrial BSs $\mathcal{B} = \{1, \ldots, B\}$, a constellation of LEO satellites $\mathcal{S} = \{1, \ldots, S\}$, and a remote cloud center C. Each UE $i \in \mathcal{U}$ continuously generates computation tasks that require joint communication and processing resources. A task is characterized by three parameters: the input data size d_i (in bits), the computation intensity c_i (in CPU cycles per bit), and the maximum tolerable latency $T_i^{\max}$ (in seconds).

The network follows a hierarchical computation model aligned with common IoT deployments:

1) **Device layer**, where each UE is capable of local computation through its embedded CPU with limited computing frequency f_i^{U}.
2) **Edge layer**, where terrestrial BSs provide access to a satellite-assisted MEC platform. Each satellite $s \in \mathcal{S}$ is equipped with an edge server of capacity f_s^{S} (in cycles/s), acting as an intermediate computation node capable of handling delay-sensitive IoT workloads while maintaining wide coverage for remote UEs.
3) **Cloud layer**, representing a centralized cloud data center with abundant computing resources f^{C}, which can process highly intensive tasks but typically introduces long end-to-end transmission delay.

Let $\mathcal{N}$ denote the set of candidate execution nodes: $\mathcal{N} \triangleq \mathcal{U} \cup \mathcal{S} \cup \mathcal{C}$, where $\mathcal{U}$ corresponds to local execution at the UE, $\mathcal{S}$ represents execution at the set of satellites, and $\mathcal{C}$ denotes the cloud center that handles this task generated by the UE. Each UE can determine where to execute its task according to system load, link quality, and latency tolerance.

To enable flexible task partitioning, we define the offloading ratio $\beta_i \in [0, 1]$ for each UE i. When $\beta_i = 0$, the entire task is computed locally; when $\beta_i = 1$, the task is fully offloaded for remote execution. Partial offloading ($0 < \beta_i < 1$) is also supported, allowing a portion of the task to be processed at the UE while the remainder is transmitted to the satellite or cloud. The system thereby achieves a trade-off between computational efficiency and communication overhead, enabling adaptive resource utilization under various IoT network conditions.

B. Propagation and Channel Gains

Accurate modeling of wireless propagation is essential for evaluating both transmission delay and energy consumption in the hybrid ground–satellite system. The network involves three uplink segments—user-to-BS (U2B), user-to-satellite (U2S), and BS-to-satellite (B2S)—as well as the satellite-to-cloud backhaul (S2C). These links differ in propagation distance, frequency band, and LoS probability, thus exhibiting distinct attenuation and latency characteristics.

For the terrestrial U2B link, each user equipment (UE) $i \in \mathcal{U}$ connects to one BS $j \in \mathcal{B}$ that provides coverage. The large-scale attenuation mainly depends on the UE–BS distance $d_{i,j}^{\mathrm{UB}}$ and environmental factors, following the conventional log-distance path loss model:

$$\mathrm{PL}_{i,j}^{\mathrm{UB}} = \mathrm{PL}_0 + 10\,\gamma \log_{10}\left(\frac{d_{i,j}^{\mathrm{UB}}}{d_0}\right), \tag{1}$$

where PL_0 is the reference loss at distance d_0 and γ denotes the path-loss exponent determined by the propagation environment [30].

In contrast, satellite-related segments (U2S, B2S, and S2C) are dominated by free-space propagation. With satellite index $s \in \mathcal{S}$ and the corresponding slant distances $d_{i,s}^{\mathrm{US}}$, $d_{j,s}^{\mathrm{BS}}$, and d_s^{SC}, the loss mainly depends on d and can be modeled as

$$\mathrm{PL}^{\mathrm{FS}}(d) = 20\log_{10} d + 20\log_{10} f + 20\log_{10}\left(\frac{4\pi}{c}\right), \tag{2}$$

where f is the carrier frequency and c is the speed of light. Substituting $d = d_{i,s}^{\mathrm{US}}$, $d = d_{j,s}^{\mathrm{BS}}$, and $d = d_s^{\mathrm{SC}}$ yields the respective path losses for the UE–satellite, BS–satellite, and satellite–cloud links via satellite s.

For analytical consistency, each channel is expressed in terms of its linear power gain obtained from the path loss (in dB):

$$g_{i,j}^{\mathrm{UB}} = 10^{-\mathrm{PL}_{i,j}^{\mathrm{UB}}/10}, \tag{3}$$

$$g_{i,s}^{\mathrm{US}} = 10^{-\mathrm{PL}^{\mathrm{FS}}(d_{i,s}^{\mathrm{US}})/10}, \tag{4}$$

$$g_{j,s}^{\mathrm{BS}} = 10^{-\mathrm{PL}^{\mathrm{FS}}(d_{j,s}^{\mathrm{BS}})/10}, \tag{5}$$

$$g_{s}^{\mathrm{SC}} = 10^{-\mathrm{PL}^{\mathrm{FS}}(d_{s}^{\mathrm{SC}})/10}. \tag{6}$$

These gains characterize the large-scale channel conditions used in the rate, delay, and energy computations throughout the system analysis.

Finally, fixed propagation latencies are included to capture signal travel time over long distances. Let $\tau_{i,s}$, $\tau_{j,s}$, and $\tau_{s,C}$ denote the propagation delays of the UE i–satellite s, BS j–satellite s, and satellite s–cloud links, respectively. These constants are added to transmission times when evaluating end-to-end offloading delay.

C. Uplink Access and Achievable Rates

To reflect architectural roles for IoT uplinks, we organize uplink communication into two categories: ground-to-ground (G2G) access and ground–space (G2S) communications, including UE–satellite (U2S), BS–satellite (B2S), and satellite–cloud (S2C). The receiver noise power over bandwidth B is expressed as $\sigma^2 = N_0 B$, where N_0 denotes the one-sided thermal noise power spectral density.

For the G2G uplink (U2B) with power-domain NOMA and SIC, multiple UEs may share BS j on the same time–frequency resource. Let $\mathcal{U}_j$ denote the UE set scheduled at j, and $\pi_j : \mathcal{U}_j \to \{1, \ldots, |\mathcal{U}_j|\}$ be the SIC decoding order [31]. The set of uncanceled interferers when decoding UE i is $\mathcal{U}_j^{\succ i} \triangleq \{\, k \in \mathcal{U}_j : \ \pi_j(k) > \pi_j(i) \,\}$, where $\mathcal{U}_j^{\succ i}$ represents the set of uncanceled interferers when decoding UE i. Allowing per-UE powers $P_{u,i}$ and bandwidth B_{UB}, the achievable rate of UE $i \in \mathcal{U}_j$ is

$$\begin{aligned} R_{i,j}^{\mathrm{UB}} &= B_{\mathrm{UB}} \log_2\!\left(1 + \frac{P_{u,i}\, g_{i,j}^{\mathrm{UB}}}{\displaystyle\sum_{k \in \mathcal{U}_j^{\succ i}} P_{u,k}\, g_{k,j}^{\mathrm{UB}} + \sigma_{\mathrm{UB}}^2}\right), \\ \sigma_{\mathrm{UB}}^2 &= N_0 B_{\mathrm{UB}}. \end{aligned} \tag{7}$$

Here, UE i treats only later-decoded signals as interference; earlier-decoded signals are canceled via SIC.

For the G2S links, we assume orthogonalization among the U2S, B2S, and S2C segments so that intra-segment interference is negligible. These segments form a space communication chain through which data can be transmitted directly from UEs to the satellite or relayed via BSs before reaching the satellite and the remote cloud. Each rate depends on the corresponding bandwidth, transmit power, and channel gain.

The achievable rate of the UE–satellite (U2S) link under bandwidth B_{US} and UE transmit power $P_{u,i}$ is given by

$$R_{i,s}^{\mathrm{US}} = B_{\mathrm{US}}\, \log_2\!\left(1 + \frac{P_{u,i}\, g_{i,s}^{\mathrm{US}}}{\sigma_{\mathrm{US}}^2}\right), \qquad \sigma_{\mathrm{US}}^2 = N_0 B_{\mathrm{US}}. \tag{8}$$

This represents the achievable data rate between UE i and satellite s, where $g_{i,s}^{\mathrm{US}}$ captures the corresponding channel gain and σ_{US}^2 denotes the receiver noise power.

The BS–satellite (B2S) link is modeled similarly. With BS transmit power $P_{b,j}$ and bandwidth B_{BS}, its achievable rate is

$$R_{j,s}^{\mathrm{BS}} = B_{\mathrm{BS}}\, \log_2\!\left(1 + \frac{P_{b,j}\, g_{j,s}^{\mathrm{BS}}}{\sigma_{\mathrm{BS}}^2}\right), \qquad \sigma_{\mathrm{BS}}^2 = N_0 B_{\mathrm{BS}}. \tag{9}$$

This rate quantifies the data throughput of BS j when forwarding offloaded tasks to satellite s.

For the satellite–cloud (S2C) backhaul, we model the transmission rate at the same physical-layer level. With satellite transmit power P_s, bandwidth B_{SC}, and average gain g_s^{SC}, the achievable rate is

$$R_{s}^{\mathrm{SC}} = B_{\mathrm{SC}}\, \log_2\!\left(1 + \frac{P_{s}\, g_{s}^{\mathrm{SC}}}{\sigma_{\mathrm{SC}}^2}\right), \qquad \sigma_{\mathrm{SC}}^2 = N_0 B_{\mathrm{SC}}. \tag{10}$$

This rate captures the physical-layer capacity of the satellite–cloud segment, determined by satellite transmission power and the free-space channel gain.

D. Execution Modes, Latency, and Energy

We consider *partial offloading*, where a fraction $(1 - \beta_i)$ of task i is executed locally on the UE and the remaining fraction $\beta_i \in [0, 1]$ is offloaded to a remote node. Computation is performed only at the UE (local), the satellite MEC, and the cloud; the BS acts purely as a relay. Hence there are five end-to-end paths: UE Local, UE–BS–Satellite, UE–Satellite, UE–BS–Satellite–Cloud, and UE–Satellite–Cloud. For partial offloading, the local and remote parts run concurrently, so each path latency equals the maximum of the local-computation time and the corresponding remote-path delay.

1) Local Path (UE Local): The local computation latency is

$$T_i^{\mathrm{U}} = \frac{(1 - \beta_i)\, d_i c_i}{f_i^{\mathrm{U}}}, \tag{11}$$

where d_i is input size, c_i is per-bit cycle demand, and f_i^{U} is the UE CPU frequency.

The local-computation energy follows the dynamic power model

$$E_i^{\mathrm{U,comp}} = \kappa_i (1 - \beta_i) d_i c_i \big(f_i^{\mathrm{U}}\big)^2, \tag{12}$$

where κ_i is the effective switching-capacitance coefficient of UE i.

2) Satellite Path via BS Relay (UE–BS–Satellite): When the offloaded fraction goes U→B→S and is computed at the satellite, the transmission delay is

$$T_{i,j}^{\mathrm{S,tx—U2B2S}} = \frac{\beta_i d_i}{R_{i,j}^{\mathrm{UB}}} + \tau_{j,s} + \frac{\beta_i d_i}{R_{j,s}^{\mathrm{BS}}}, \tag{13}$$

where $\tau_{j,s}$ is the B2S propagation delay between BS j and satellite s, and $R_{i,j}^{\mathrm{UB}}$, $R_{j,s}^{\mathrm{BS}}$ are U2B and B2S rates.

The satellite computation time is

$$T_{i,s}^{\mathrm{S,comp}} = \frac{\beta_i d_i c_i}{f_s^{\mathrm{S}}}, \tag{14}$$

where f_s^{S} denotes the computing frequency of satellite s. The remote-path delay is $T_{i,j,s}^{\mathrm{S—U2B2S}} = T_{i,j,s}^{\mathrm{S,tx—U2B2S}} + T_{i,s}^{\mathrm{S,comp}}$, and the corresponding end-to-end latency is

$$T_{i,j,s}^{(\mathrm{U–B–S})} = \max\!\big\{\, T_i^{\mathrm{U}},\ T_{i,j,s}^{\mathrm{S—U2B2S}} \,\big\}. \tag{15}$$

The UE and BS transmission energies are

$$E_{i,j}^{\mathrm{UB,tx}} = P_{u,i}\,\frac{\beta_i d_i}{R_{i,j}^{\mathrm{UB}}}, \tag{16}$$

$$E_{j,s}^{\mathrm{BS,tx}} = P_{b,j}\,\frac{\beta_i d_i}{R_{j,s}^{\mathrm{BS}}}, \tag{17}$$

where $P_{u,i}$ and $P_{b,j}$ are the transmit powers of UE i and BS j. The satellite computation energy adopts the same dynamic-power form:

$$E_{i,s}^{\mathrm{S,comp}} = \kappa_s^{\mathrm{S}}\,\beta_i d_i c_i \big(f_s^{\mathrm{S}}\big)^2, \tag{18}$$

where κ_s^{S} is the effective switching-capacitance coefficient of satellite s.

3) Direct Satellite Path (UE–Satellite): For direct U→S transmission with satellite computing, the transmission delay is

$$T_{i,s}^{\mathrm{S,tx—U2S}} = \tau_{i,s} + \frac{\beta_i d_i}{R_{i,s}^{\mathrm{US}}}, \tag{19}$$

where $\tau_{i,s}$ is the U2S propagation delay and $R_{i,s}^{\mathrm{US}}$ is the U2S rate. The remote-path delay is $T_{i,s}^{\mathrm{S—U2S}} = T_{i,s}^{\mathrm{S,tx—U2S}} + T_{i,s}^{\mathrm{S,comp}}$, yielding

$$T_i^{(\mathrm{U–S})} = \max\big\{\, T_i^{\mathrm{U}},\ T_i^{\mathrm{S—U2S}} \,\big\}. \tag{20}$$

The UE transmission energy for U2S is

$$E_{i,s}^{\mathrm{US,tx}} = P_{u,i}\,\frac{\beta_i d_i}{R_{i,s}^{\mathrm{US}}}, \tag{21}$$

and the satellite computation energy is given in (18).

4) Cloud Path via BS and Satellite (UE–BS–Satellite–Cloud): For U→B→S→Cloud with cloud computing, the transmission delay is

$$T_{i,j,s}^{\mathrm{C,tx—U2B2S2C}} = \tau_{j,s} + \tau_{s,C} + \frac{\beta_i d_i}{R_{i,j}^{\mathrm{UB}}} + \frac{\beta_i d_i}{R_{j,s}^{\mathrm{BS}}} + \frac{\beta_i d_i}{R_s^{\mathrm{SC}}}, \tag{22}$$

where $\tau_{s,C}$ and R_s^{SC} are the S2C propagation delay and rate.

The cloud computation time is

$$T_i^{\mathrm{C,comp}} = \frac{\beta_i d_i c_i}{f^{\mathrm{C}}}, \tag{23}$$

with f^{C} the cloud CPU frequency. The remote-path delay is $T_{i,j,s}^{\mathrm{C—U2B2S2C}} = T_{i,j,s}^{\mathrm{C,tx—U2B2S2C}} + T_i^{\mathrm{C,comp}}$, and the overall latency is

$$T_{i,j,s}^{(\mathrm{U–B–S–C})} = \max\big\{\, T_i^{\mathrm{U}},\ T_{i,j,s}^{\mathrm{C—U2B2S2C}} \,\big\}. \tag{24}$$

The transmission energies on the three links are given by (16), (17), and

$$E_s^{\mathrm{SC,tx}} = P_s\,\frac{\beta_i d_i}{R_s^{\mathrm{SC}}}, \tag{25}$$

where P_s is the satellite-side transmit power on the S2C link. The cloud computation energy uses the same dynamic-power form:

$$E_i^{\mathrm{C,comp}} = \kappa^{\mathrm{C}}\,\beta_i d_i c_i \big(f^{\mathrm{C}}\big)^2, \tag{26}$$

where κ^{C} is the cloud-side switching-capacitance coefficient.

5) Direct Cloud Path via Satellite (UE–Satellite–Cloud): For the direct U→S→Cloud route,

$$T_{i,s}^{\mathrm{C,tx—U2S2C}} = \tau_{i,s} + \tau_{s,C} + \frac{\beta_i d_i}{R_{i,s}^{\mathrm{US}}} + \frac{\beta_i d_i}{R_s^{\mathrm{SC}}}, \tag{27}$$

and the remote-path delay is $T_{i,s}^{\mathrm{C—U2S2C}} = T_{i,s}^{\mathrm{C,tx—U2S2C}} + T_i^{\mathrm{C,comp}}$, yielding

$$T_{i,s}^{(\mathrm{U–S–C})} = \max\big\{\, T_i^{\mathrm{U}},\ T_{i,s}^{\mathrm{C—U2S2C}} \,\big\}. \tag{28}$$

The transmission energies are given by (21) and (25), and the cloud computation energy by (26).

E. Problem Formulation

Let $\mathcal{P} = \{\mathrm{U}, \mathrm{UBS}, \mathrm{US}, \mathrm{UBSC}, \mathrm{USC}\}$ denote the five admissible path types (UE Local, UE–BS–Satellite, UE–Satellite, UE–BS–Satellite–Cloud, UE–Satellite–Cloud), and let $\mathcal{B}$ and $\mathcal{S}$ denote the sets of BS and satellites, respectively. For each task $i \in \mathcal{U}$, introduce the path-selection binaries $z_{i,p} \in \{0,1\}$ for $p \in \mathcal{P}$, the relay–BS selection binaries $y_{i,j,s}^{\mathrm{UBS}}, y_{i,j,s}^{\mathrm{UBSC}} \in \{0,1\}$ for $j \in \mathcal{B}, s \in \mathcal{S}$, and the direct-satellite selection binaries $s_{i,s}^{\mathrm{US}}, s_{i,s}^{\mathrm{USC}} \in \{0,1\}$ for $s \in \mathcal{S}$.

The end-to-end latency and energy of task i are selected by the binaries as

$$\begin{aligned} T_i \;=\; & z_{i,\mathrm{U}}\,T_i^{\mathrm{U}} + \sum_{j\in\mathcal{B}}\sum_{s\in\mathcal{S}} y_{i,j,s}^{\mathrm{UBS}}\,T_{i,j,s}^{(\mathrm{U–B–S})} + \sum_{s\in\mathcal{S}} s_{i,s}^{\mathrm{US}}\,T_{i,s}^{(\mathrm{U–S})} \\ & + \sum_{j\in\mathcal{B}}\sum_{s\in\mathcal{S}} y_{i,j,s}^{\mathrm{UBSC}}\,T_{i,j,s}^{(\mathrm{U–B–S–C})} + \sum_{s\in\mathcal{S}} s_{i,s}^{\mathrm{USC}}\,T_{i,s}^{(\mathrm{U–S–C})}, \end{aligned} \tag{29}$$

$$\begin{aligned} E_i \;=\; & z_{i,\mathrm{U}}\,E_i^{(\mathrm{U})} + \sum_{j\in\mathcal{B}}\sum_{s\in\mathcal{S}} y_{i,j,s}^{\mathrm{UBS}}\,E_{i,j,s}^{(\mathrm{U–B–S})} + \sum_{s\in\mathcal{S}} s_{i,s}^{\mathrm{US}}\,E_{i,s}^{(\mathrm{U–S})} \\ & + \sum_{j\in\mathcal{B}}\sum_{s\in\mathcal{S}} y_{i,j,s}^{\mathrm{UBSC}}\,E_{i,j,s}^{(\mathrm{U–B–S–C})} + \sum_{s\in\mathcal{S}} s_{i,s}^{\mathrm{USC}}\,E_{i,s}^{(\mathrm{U–S–C})}, \end{aligned} \tag{30}$$

where the per-path energies assemble previously defined components:

$$E_i^{(\mathrm{U})} = E_i^{\mathrm{U,comp}}, \tag{31}$$

$$E_{i,j,s}^{(\mathrm{U–B–S})} = E_i^{\mathrm{U,comp}} + E_{i,j}^{\mathrm{UB,tx}} + E_{j,s}^{\mathrm{BS,tx}} + E_{i,s}^{\mathrm{S,comp}}, \tag{32}$$

$$E_{i,s}^{(\mathrm{U–S})} = E_i^{\mathrm{U,comp}} + E_{i,s}^{\mathrm{US,tx}} + E_{i,s}^{\mathrm{S,comp}}, \tag{33}$$

$$E_{i,j,s}^{(\mathrm{U–B–S–C})} = E_i^{\mathrm{U,comp}} + E_{i,j}^{\mathrm{UB,tx}} + E_{j,s}^{\mathrm{BS,tx}} + E_s^{\mathrm{SC,tx}} + E_i^{\mathrm{C,comp}}, \tag{34}$$

$$E_{i,s}^{(\mathrm{U–S–C})} = E_i^{\mathrm{U,comp}} + E_{i,s}^{\mathrm{US,tx}} + E_s^{\mathrm{SC,tx}} + E_i^{\mathrm{C,comp}}. \tag{35}$$

We jointly optimize delay and energy via a weighted sum:

$$\min_{\{z,y,\beta\}} \quad \sum_{i\in\mathcal{U}} \left(\omega_T\, T_i + \omega_E\, E_i\right) \tag{36a}$$

$$\text{s.t.} \quad T_i \le T_i^{\max}, \qquad \forall\, i \in \mathcal{U}, \tag{36b}$$

$$z_{i,\mathrm{U}} + \sum_{p\in\{\mathrm{UBS},\,\mathrm{US},\,\mathrm{UBSC},\,\mathrm{USC}\}} z_{i,p} = 1, \quad \forall\, i \in \mathcal{U}, \tag{36c}$$

$$\sum_{j\in\mathcal{B}} y_{i,j}^{\mathrm{UBS}} = z_{i,\mathrm{UBS}}, \qquad \forall\, i \in \mathcal{U},$$
$$\sum_{j\in\mathcal{B}} y_{i,j,s}^{\mathrm{UBSC}} = z_{i,\mathrm{UBSC}}, \qquad \forall\, i \in \mathcal{U}, \tag{36d}$$

$$0 \le \beta_i \le 1 - z_{i,\mathrm{U}}, \qquad \forall\, i \in \mathcal{U}, \tag{36e}$$

$$z_{i,p} \in \{0,1\}, \quad y_{i,j,s}^{\mathrm{UBS}},\, y_{i,j,s}^{\mathrm{UBSC}} \in \{0,1\}, \forall\, i,p,j. \tag{36f}$$

In (29)–(36f), the per-link rates inside the latency expressions (e.g., $R_{i,j}^{\mathrm{UB}}$, $R_{j,s}^{\mathrm{BS}}$, $R_{i,s}^{\mathrm{US}}$, R_s^{SC}) and propagation delays ($\tau_{j,s}$, $\tau_{i,s}$, $\tau_{s,C}$) follow the physical-layer models given previously. The objective (36a) provides a delay–energy trade-off controlled by weights $\omega_T, \omega_E > 0$. Due to the binary variables and nonconvex rate expressions inside T_i and E_i, problem (36a)–(36f) is a MINLP.

V. Algorithm

A. Solver-Based Optimal Scheduling

We treat the optimization problem (36a)–(36f) as a MINLP formulation and develop a solver-based optimal scheduling algorithm to obtain the global optimum, serving as the benchmark for subsequent comparisons. The algorithm preserves the exact U2B NOMA–SIC rate in (7) and the orthogonal U2S/B2S/S2C links in (8)–(10), while path-wise latency and energy follow (29)–(35) with deadlines enforced by (36b). To maintain physical-layer fidelity, no full MILP reformulation is applied: the $\log(1+\mathrm{SINR})$ term is implemented via the solver's logarithmic general constraint, and bilinear equalities such as $\mathrm{SINR}_{i,j} D_{i,j} = P_{u,i} g_{i,j}^{\mathrm{UB}} x_{i,j}$ and $R_{i,j}^{\mathrm{UB}} t_{i,j}^{\mathrm{UB}} = \beta_i d_i x_{i,j}$ are retained, yielding a nonconvex MIQCP solved by Gurobi. Appropriate relative and absolute MIPGap tolerances and a wall-clock time limit are set, returning the best incumbent solution if the limit is reached.

This solver-based method achieves the true global optimum in mini-scale experiments and provides a high-quality benchmark for evaluating the efficiency and scalability of the decentralized scheduler introduced in Sec. V-B. It should be noted that the solver-based method remains capable of finding the global optimum in large-scale settings as well; however, its computational cost grows rapidly with system size, making it impractical for real-time or latency-sensitive scenarios rather than theoretically infeasible.

B. Game-Theoretic Offloading and Scheduling

Decentralized best-response is attractive for IoT-scale systems with massive device counts and intermittent links, where per-UE energy budgets and URLLC-style deadlines preclude heavy centralized optimization at run time. We therefore design a decentralized scheduler in which each UE selfishly selects an offloading *path* (and, if applicable, a relay BS) to maximize an individual benefit consistent with the latency/energy models in Sec. IV. The U2B segment is coupled by NOMA–SIC rates in (7), while U2S/B2S/S2C are orthogonal as in (8)–(10).

1) Benefit function: For user i, define the weighted delay–energy *cost*

$$J_i(a) \;\triangleq\; \omega_T\, T_i(a) \;+\; \omega_E\, E_i(a), \tag{37}$$

where a is the joint action profile; $T_i(a)$ and $E_i(a)$ are assembled from (29)–(35) using the same link rates and propagation latencies as Sec. IV. The individual *benefit* is

$$\tilde{B}_i(a) \;\triangleq\; -\,J_i(a), \tag{38}$$

so maximizing benefit is equivalent to minimizing (37).

2) Game-theoretic modeling: We model the distributed decision process as a finite non-cooperative game

$$\mathcal{G} = \langle \mathcal{U}, \{\mathcal{A}_i\}, \{\tilde{B}_i\} \rangle. \tag{39}$$

Players: all UEs $i \in \mathcal{U}$ participate as self-interested agents.

Actions: each user selects one offloading action $a_i = (p_i, b_i)$, where $p_i \in \mathcal{P} = \{\mathrm{U}, \mathrm{US}, \mathrm{UBS}, \mathrm{USC}, \mathrm{UBSC}\}$ and $b_i = \perp$ if $p_i \in \{\mathrm{U}, \mathrm{US}, \mathrm{USC}\}$; otherwise $b_i \in \mathcal{B}_i$ is a BS chosen from the coverage-limited candidate set for user i.

The interaction among players arises only through the U2B transmission, where NOMA users sharing the same BS influence each other's achievable rates in (7). For all other segments (U2S, B2S, S2C), the links are orthogonal, and their rates follow (8)–(10) independently of other users' choices.

For a given action profile a_{-i} of other users, the feasible set of user i is defined as

$$\mathcal{A}_i^{\mathrm{feas}}(a_{-i}) = \left\{ a_i \in \mathcal{A}_i \,:\, T_i(a_i, a_{-i}) \le T_i^{\max} \right\}. \tag{40}$$

We only consider candidates with $T_i, E_i < \infty$. Infeasible candidates—those causing deadline violations or non-finite link metrics—are excluded. If no feasible path exists, the user retains their previous action or falls back to local execution.

Each user aims to maximize its benefit $\tilde{B}_i$, which is equivalent to minimizing the individual weighted cost J_i in (37). Users make decisions in a myopic manner: each evaluates its own latency and energy based on the current system state a_{-i} without predicting others' future adjustments.

Definition 1 (Nash Equilibrium)**.** An action profile $a^* = (a_1^*, a_2^*, \ldots, a_{|\mathcal{U}|}^*)$ is a *Nash equilibrium (NE)* of the game $\mathcal{G} = \langle \mathcal{U}, \{\mathcal{A}_i\}, \{J_i\} \rangle$ if no user can further reduce its individual cost by unilaterally changing its own strategy while others keep theirs fixed, i.e.,

$$J_i(a_i^*, a_{-i}^*) \;\le\; J_i(a_i, a_{-i}^*), \qquad \forall\, a_i \in \mathcal{A}_i,\; i \in \mathcal{U}. \tag{41}$$

Equivalently, when expressed in terms of the benefit function $\tilde{B}_i = -J_i$, the NE condition can be written as

$$\tilde{B}_i(a_i^*, a_{-i}^*) \;\geq\; \tilde{B}_i(a_i, a_{-i}^*), \qquad \forall\, a_i \in \mathcal{A}_i,\; i \in \mathcal{U}. \quad (42)$$

At equilibrium, each user's offloading decision is its best response to the strategies of the other $|\mathcal{U}| - 1$ users. The existence of at least one NE is crucial, since it guarantees that the distributed offloading process reaches a stable operating point where no user has an incentive to deviate unilaterally.

Definition 2 (Potential Game)**.** A finite game $\mathcal{G} = \langle \mathcal{U}, \{\mathcal{A}_i\}, \{J_i\} \rangle$ is an *OPG* if there exists a real-valued function $\Phi(a)$, referred to as the *potential function*, such that for every user $i \in \mathcal{U}$, for all feasible actions $a_i, a_i' \in \mathcal{A}_i$ and all $a_{-i} \in \prod_{k \neq i} \mathcal{A}_k$, the following implication holds:

$$J_i(a_i', a_{-i}) < J_i(a_i, a_{-i}) \;\Rightarrow\; \Phi(a_i', a_{-i}) < \Phi(a_i, a_{-i}). \quad (43)$$

In other words, any unilateral improvement in an individual player's cost leads to a strict decrease in the global potential function.

Based on (41), any Nash equilibrium of our game coincides with a unilateral local minimum of the potential function $\Phi(a)$ (i.e., no unilateral deviation can further decrease Φ). For the considered offloading game, define for each BS j the U2B user set $\mathcal{U}_j(a) = \{\, i \in \mathcal{U} :$ action a_i uses U2B via $j \,\}$ and a deterministic SIC order π_j (e.g., sorted by large-scale gain, where a larger $\pi_j(i)$ indicates that user i is decoded *later*). Let the residual-interference aggregator be

$$I_j(a) \triangleq \sum_{k \in \mathcal{U}_j(a)} \sum_{\ell \in \mathcal{U}_j(a):\, \pi_j(\ell) > \pi_j(k)} \phi\big(P_{u,\ell}\, g_{\ell j}^{\mathrm{UB}}\big),$$

where $\phi(\cdot)$ is a non-decreasing function, indicating the aggregation of the "excess contributions" of user ℓ over the residual contribution of user k after decoding. Define

$$\Psi_j(a) \triangleq -\varpi\, I_j(a), \qquad \varpi \;\geq\; \sup_a \max_{j \in \mathcal{B}} \sum_{m \in \mathcal{U}_j(a)} \kappa_{m,j},$$

and denote the potential function as:

$$\Phi(a) \;=\; \sum_{i \in \mathcal{U}} J_i(a) \;+\; \sum_{j \in \mathcal{B}} \Psi_j(a). \quad (44)$$

Here, the first term $\sum_{i \in \mathcal{U}} J_i(a)$ represents the total cost of all users, where $J_i(a)$ is the cost function of user i, capturing the weighted delay and energy consumption. The second term $\sum_{j \in \mathcal{B}} \Psi_j(a)$ accounts for the interference at each BS, where $\Psi_j(a)$ reflects the residual interference $I_j(a)$ at BS j, which negatively impacts the overall system performance.

Additionally, $\kappa_{m,j}$ denotes a finite Lipschitz constant that upper-bounds the sensitivity of user m's individual cost J_m with respect to the residual interference at BS j. In other words, it quantifies how much the cost function J_m changes when the interference level I_j at BS j varies. A detailed explanation of $\kappa_{m,j}$ can be found in Lemma 1.

Lemma 1 (Bounded sensitivity of J_m to residual interference)**.** *Fix any BS j. Under bounded parameters (finite $|\mathcal{U}_j|$, $P_{u,i} \leq P_u^{\max}$, $g_{ij}^{\mathrm{UB}} \leq g^{\max}$, positive bandwidths and $\sigma^2 > 0$), there exist finite nonnegative constants $\{\kappa_{m,j}\}_{m \in \mathcal{U}_j}$ such that for any two profiles differing only in the U2B membership/order at j,*

$$\sum_{m \in \mathcal{U}_j} \big(J_m(a') - J_m(a)\big) \;\leq\; \Big(\sum_{m \in \mathcal{U}_j} \kappa_{m,j}\Big)\big(I_j(a') - I_j(a)\big). \quad (45)$$

Proof. We aim to prove that the cost J_m of user m is bounded in its sensitivity to the residual interference I_j at base station j. Specifically, we will show that the change in J_m due to variations in I_j is bounded.

We focus on the U2B transmission terms of J_m, as the cost is primarily influenced by this term. The U2B transmission rate for user m, denoted by $R_{m,j}^{\mathrm{UB}}$, is given by the expression:

$$R_{m,j}^{\mathrm{UB}} = B \log_2\left(1 + \frac{S_{m,j}}{I_{m,j} + \sigma^2}\right),$$

where $S_{m,j}$ represents the signal power from user m to base station j, defined as $S_{m,j} = P_{u,m} g_{mj}^{\mathrm{UB}}$, with $P_{u,m}$ being the transmission power and g_{mj}^{UB} being the channel gain. The term $I_{m,j}$ is the residual interference at base station j, and σ^2 represents the noise power.

Next, we analyze the sensitivity of the transmission rate $R_{m,j}^{\mathrm{UB}}$ with respect to the interference $I_{m,j}$. Taking the derivative of the rate expression with respect to $I_{m,j}$, we obtain:

$$\frac{\partial R_{m,j}^{\mathrm{UB}}}{\partial I_{m,j}} = \frac{B}{\ln 2} \cdot \frac{-S_{m,j}}{(I_{m,j} + \sigma^2)\,(I_{m,j} + \sigma^2 + S_{m,j})}.$$

This derivative describes how the rate changes as the interference $I_{m,j}$ varies. The term is always negative because increasing the interference reduces the achievable rate.

Now, since the delay T_m^{UB} and energy E_m^{UB} are inversely proportional to the transmission rate $R_{m,j}^{\mathrm{UB}}$, we know that:

$$T_m^{\mathrm{UB}} \propto \frac{1}{R_{m,j}^{\mathrm{UB}}} \quad \text{and} \quad E_m^{\mathrm{UB}} \propto \frac{1}{R_{m,j}^{\mathrm{UB}}}.$$

Thus, any change in the rate $R_{m,j}^{\mathrm{UB}}$ will cause a corresponding change in the delay and energy costs. Using the mean-value theorem, we can express the change in J_m as:

$$|J_m(a') - J_m(a)| \leq \kappa_{m,j} \cdot |I_{m,j}(a') - I_{m,j}(a)|,$$

where $\kappa_{m,j}$ is a constant that depends on the global bounds, such as the maximum values of the power $P_u^{\max}$, the channel gain $g^{\max}$, and the weights ω_T and ω_E. This constant $\kappa_{m,j}$ bounds how much the cost J_m can change given a change in the interference.

Finally, we aggregate the changes across all users in the SIC chain at base station j, leading to the inequality:

$$\sum_{m \in \mathcal{U}_j} (J_m(a') - J_m(a)) \leq \left(\sum_{m \in \mathcal{U}_j} \kappa_{m,j}\right) \cdot (I_j(a') - I_j(a)).$$

This result shows that the total change in the cost function J_m for all users at base station j is bounded by the change in the residual interference I_j, scaled by a constant that depends

on the system parameters. This proves the bounded sensitivity of the cost to the interference, as desired. □

Theorem 2 (Ordinal potential property)**.** *For the game $\mathcal{G}$ defined above, if player i unilaterally changes from a_i to a_i' such that $J_i(a_i', a_{-i}) < J_i(a_i, a_{-i})$, then the potential function Φ strictly decreases, i.e., $\Phi(a_i', a_{-i}) < \Phi(a_i, a_{-i})$.*

Proof. By Lemma 1, for each $j \in \mathcal{B}$,

$$\sum_{m \in \mathcal{U}_j} \Delta J_m \;\le\; \Big(\sum_{m \in \mathcal{U}_j} \kappa_{m,j} \Big) \Delta I_j, \qquad \Delta \Psi_j = -\,\varpi\, \Delta I_j.$$

Hence,

$$\sum_{m \in \mathcal{U}_j} \Delta J_m + \Delta \Psi_j \;\le\; \Big(\textstyle\sum_{m \in \mathcal{U}_j} \kappa_{m,j} - \varpi \Big) \Delta I_j \;\le\; 0$$

whenever $\varpi \ge \max_{j \in \mathcal{B}} \sum_{m \in \mathcal{U}_j} \kappa_{m,j}$. Summing over j gives $\sum_{m \neq i} \Delta J_m + \sum_j \Delta \Psi_j \le 0$, Since $\Delta J_i < 0$ by the strictly improving deviation, we have $\Delta \Phi = \Delta J_i + \sum_{m \neq i} \Delta J_m + \sum_j \Delta \Psi_j < 0$. □

This establishes that $\mathcal{G}$ is an *OPG*. Consequently, the best-response dynamics (BRD) monotonically decreases Φ and converges in a finite number of steps to a pure-strategy Nash equilibrium.

Corollary 3 (Finite-step convergence of BRD)**.** *Under strict-improvement best responses, Φ strictly decreases at every update. Since the action space is finite (or, with hard deadlines, the feasible subspace is finite), BRD converges in finitely many steps to a pure-strategy Nash equilibrium.*

3) Algorithm Description: The algorithm shown in Algorithm 1 begins by taking the necessary inputs (line 1) and initializing each user's decision (line 2).

The algorithm enters a loop where the round counter is incremented, and a flag `changed` is set to false (line 4). It then proceeds with a random permutation of users and evaluates each user's offloading candidates depending on the base stations $j \in \mathcal{B}_i$ (line 6). For each candidate action (p, j), it computes the rates, delays, and energy and updates the objective function J_i^λ with the penalty if the delay exceeds the maximum threshold $T_i^{\max}$ (lines 7-8).

If $\lambda = 0$, it discards any candidates with non-finite values for delay or energy (line 9). The best offloading action is selected by minimizing the cost function J_i^λ, breaking ties first by fixed path priority and then by the smallest base station index (line 10). If the selected path differs from the user's current decision, the algorithm updates the user's choice and recalculates the SIC order if necessary (line 13). The flag `changed` is set to true.

The algorithm continues iterating until no changes are made during a round, or the round cap $R_{\max}$ is reached (line 12). In the final pass, the algorithm recomputes the metrics for all users and writes back the total cost and benefits (line 17).

4) Convergence Analysis: Let $a^{(t)}$ be the joint action profile after the t-th update in Algorithm 1, and let $\Phi(a)$ be the potential. We adopt the strict-improvement rule: a user updates only if its cost strictly decreases. The SIC order at each BS is recomputed whenever its U2B membership changes, so $\Psi_j(a)$ remains deterministic.

Algorithm 1 DGTS

1: **Input:** users $\mathcal{U}$; BSs $\mathcal{B}$; coverage sets $\{\mathcal{B}_i\}$; weights (ω_T, ω_E); deadlines $\{T_i^{\max}\}$; penalty $\lambda \ge 0$; round cap $R_{\max}$; deterministic SIC rule π.
2: **Init:** $(d_i, b_i) \leftarrow (\mathrm{U}, \perp)$ for all i; if $\lambda = 0$ run a feasible-seeding heuristic; set $r \leftarrow 0$.
3: **repeat**
4: $\quad r \leftarrow r + 1$; `changed` $\leftarrow$ **false**
5: $\quad$ **for each** i in a random permutation of $\mathcal{U}$ **do**
6: $\quad\quad \widehat{\mathcal{A}}_i \leftarrow \{(\mathrm{U}, \perp), (\mathrm{US}, \perp)\} \cup \{(\mathrm{UBS}, j)\}_{j \in \mathcal{B}_i} \cup \{(\mathrm{USC}, \perp)\} \cup \{(\mathrm{UBSC}, j)\}_{j \in \mathcal{B}_i}$
7: $\quad\quad$ **for** each candidate $(p, j) \in \widehat{\mathcal{A}}_i$: compute rates by (7)–(10); assemble T_i, E_i via (29)–(35); set $J_i^\lambda \leftarrow J_i + \lambda[T_i - T_i^{\max}]_+$
8: $\quad\quad$ **if** $\lambda = 0$ **then**
9: $\quad\quad\quad$ discard candidates with non-finite (T_i, E_i) or with $T_i > T_i^{\max}$
10: $\quad\quad$ **end if**
11: $\quad\quad$ pick $(p^\star, j^\star) = \arg\min J_i^\lambda$ (ties broken by fixed path priority, then smallest BS index)
12: $\quad\quad$ **if** $(p^\star, j^\star) \neq (d_i, b_i)$ **then**
13: $\quad\quad\quad$ $(d_i, b_i) \leftarrow (p^\star, j^\star)$; update π deterministically if U2B membership at some BS changes; `changed` $\leftarrow$ **true**
14: $\quad\quad$ **end if**
15: $\quad$ **end for**
16: **until** $\neg$`changed` or $r = R_{\max}$
17: **Final pass:** recompute metrics for (d_i, b_i) for all i and write back totals, costs, and benefits

Lemma 4 (Monotonicity)**.** *If user i updates at step t with $J_i^\lambda(a_i^{(t+1)}, a_{-i}^{(t)}) < J_i^\lambda(a_i^{(t)}, a_{-i}^{(t)})$, then*

$$\Phi(a^{(t+1)}) < \Phi(a^{(t)}).$$

Lemma 5 (Fixed-point characterization)**.** *If a round ends with no user update, the resulting profile $\bar{a}$ is a pure-strategy Nash equilibrium (with respect to the feasible subspace if $\lambda = 0$).*

Theorem 6 (Convergence of DGTS)**.** *Under the strict-improvement rule, Algorithm 1 converges to a pure-strategy Nash equilibrium in a finite number of steps. If $\lambda > 0$ (soft deadlines), convergence occurs over the full action space. If $\lambda = 0$ (hard deadlines), it happens within the finite feasible subspace. The number of updates is bounded by the number of distinct potential values.*

Proof. By Lemma 4, each update strictly decreases Φ. Since the state space is finite, the process must stop after a finite number of updates. By Lemma 5, the final profile is a pure-strategy NE. The number of updates is bounded by the distinct potential values. Note that $\Phi(a)$ is bounded below. □

Discussion. Randomized user order does not affect convergence. When $\lambda = 0$, a feasible-seeding heuristic ensures the algorithm stays within the feasible subspace. Convergence

TABLE I
SIMULATION PARAMETERS AND DEFAULT VALUES.

Parameter	Description	Value
N_{cand}	Max candidate BSs per UE	10
R_{BS}	BS coverage radius	500 m-8000 m
f_{car}	Carrier frequency	2.0 GHz
α	Path-loss exponent	3.0
σ_{shad}	Log-normal shadowing deviation	8 dB
N_0	Noise power spectral density	4×10^{-21} W/Hz
$W_{\text{U2B}}^{\text{4G}}$	UE→BS bandwidth (4G)	10 MHz
$W_{\text{U2B}}^{\text{5G}}$	UE→BS bandwidth (5G)	100 MHz
W_{M2A}	UE→Satellite bandwidth	100 MHz
P_u	UE transmit power	0.1 W
P_b	BS-Satellite transmit power	1.0 W
P_s	Satellite transmit power	2.0 W
H_{sat}	Satellite altitude	780 km
C	CPU cycles required per bit	100–500 cycles/bit
$T_{\max}$	Maximum tolerable delay	10 s
f_i	UE CPU frequency	1 GHz
f_j	Satellite CPU frequency	10 GHz
f_c	Cloud CPU frequency	200 GHz
κ_i	Dynamic power coefficient (device)	1×10^{-28}
κ_j	Dynamic power coefficient (satellite)	5×10^{-28}
κ_c	Dynamic power coefficient (cloud)	1×10^{-27}
(ω_T, ω_E)	Weights of delay and energy	$(1, 10^{-3})$

occurs before the round cap due to strict potential function decrease.

VI. PERFORMANCE EVALUATION

A. Experimental Settings

All experiments were conducted on a workstation equipped with an Intel i5–14600KF CPU, 32 GB DDR5 RAM, and an NVIDIA RTX 4080 SUPER GPU (16 GB GDDR6X). The software environment includes Python 3.9 (Anaconda), PyTorch 2.7.0 with CUDA 11.8, and NumPy 1.26. The simulation parameters are summarized in Table I [28], [32]. The number of UEs varies across different experimental scales, and each UE may be associated with at most 10 BSs to keep the local search space tractable. The default ranges mirror common IoT classes: URLLC-like tasks use tighter $T_{\max}$, while mMTC-like tasks relax deadlines.

Five algorithms are evaluated:

- **COCS**: our global optimal mixed-integer nonlinear programming solution, suitable for delay-insensitive tasks but computationally heavy.
- **DGTS**: our proposed game-theoretic scheduler using BRD, targeting delay-sensitive tasks requiring rapid decisions.
- **CTPS**: a heuristic priority-based method balancing transmission and processing resources [33].
- **Local**: all tasks are executed locally on each user equipment without any offloading. This represents the conventional computation mode widely adopted in current mobile systems, It serves as the baseline reflecting the performance of traditional on-device execution in the absence of collaborative or networked computing.

TABLE II
MINI-SCALE PERFORMANCE COMPARISON AMONG ALL ALGORITHMS.

Algorithm	Avg. Cost			Avg. Delay (s)			Avg. Energy (J)		
	Mean	Std	Rank	Mean	Std	Rank	Mean	Std	Rank
COCS	0.268	0.102	1	0.179	0.066	1	86.9	36.8	1
DGTS	0.268	0.102	1	0.179	0.066	1	86.9	36.8	1
CTPS	0.315	0.128	3	0.213	0.084	3	95.7	39.5	3
Local	0.574	0.219	4	0.390	0.150	4	160.5	59.3	4
Random	0.812	0.324	5	0.551	0.218	5	189.2	70.1	5

- **Random**: random association and offloading strategy.

Two experimental regimes are studied. The mini-scale setup runs all five algorithms, including the exact COCS, to verify optimality. The large-scale setup omits COCS due to runtime constraints, comparing DGTS against CTPS, Local, and Random. Each result is averaged over 20 randomized runs. We thus focus on three core metrics: Average Cost, Average Delay, and Average Energy [27], [34].

B. Mini-Scale Results

Table II compares the five algorithms under mini-scale settings (UEs: 10–50, task size: 300–1100 KB). Each value represents the mean and standard deviation over 20 randomized runs.

The joint cost metric, which combines delay and energy using $(\omega_T, \omega_E) = (1, 10^{-3})$, clearly highlights the optimality of DGTS. As shown in Table II, DGTS achieves an average cost of 0.268, identical to that of the COCS solver, confirming convergence to the true optimal equilibrium. The CTPS heuristic follows with 0.315, roughly 17.5% higher than DGTS, indicating the loss due to its myopic resource allocation. Local and Random policies perform much worse (0.574 and 0.812, respectively), showing $2.1\times$–$3.0\times$ higher cost caused by either lack of offloading or inefficient association. The small standard deviations across all methods demonstrate stable behavior under varying small-scale conditions.

Delay statistics exhibit the same hierarchy. DGTS and COCS both reach an average latency of 0.179 s, establishing that the game-theoretic scheduler captures global-optimal delay balancing between communication and computation. CTPS increases delay to 0.213 s ($\approx$19% higher), mainly because its heuristic scheduling cannot fully coordinate between shared BSs and satellite load. Local execution has an average of 0.390 s, more than double the delay of DGTS. The Random baseline reaches 0.551 s, nearly triple DGTS's latency, due to frequent association mismatches and low spectral utilization. Overall, the results confirm that DGTS maintains near-optimal delay performance even as the number of UEs grows.

Energy consumption follows similar trends. DGTS and COCS achieve the lowest energy at 86.9 J per task, leveraging balanced offloading to satellites and cloud with efficient power allocation. CTPS consumes 95.7 J, roughly 10.1% higher, reflecting its suboptimal transmit power and task placement strategy. Local execution incurs 160.5 J on average, while Random exceeds 189 J, representing a 117% increase relative to DGTS. These results confirm that DGTS achieves energy efficiency comparable to the optimal solver, while significantly outperforming heuristic and non-cooperative baselines.

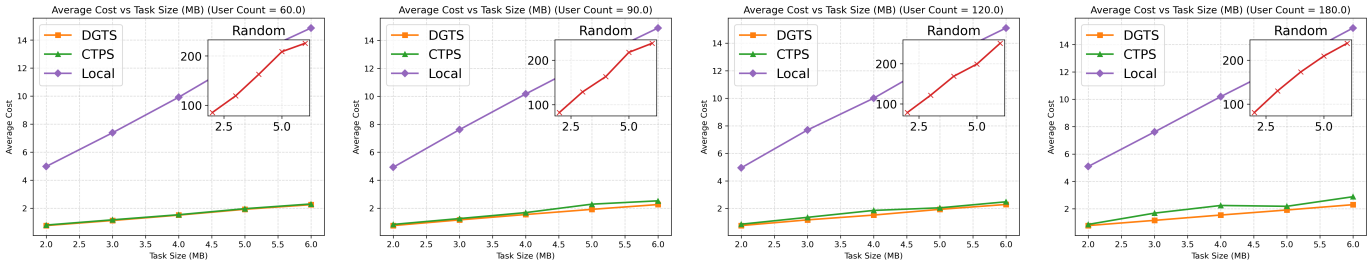

Fig. 2. Average cost versus task size for different UE counts (60, 90, 120, 180) under IoT workloads.

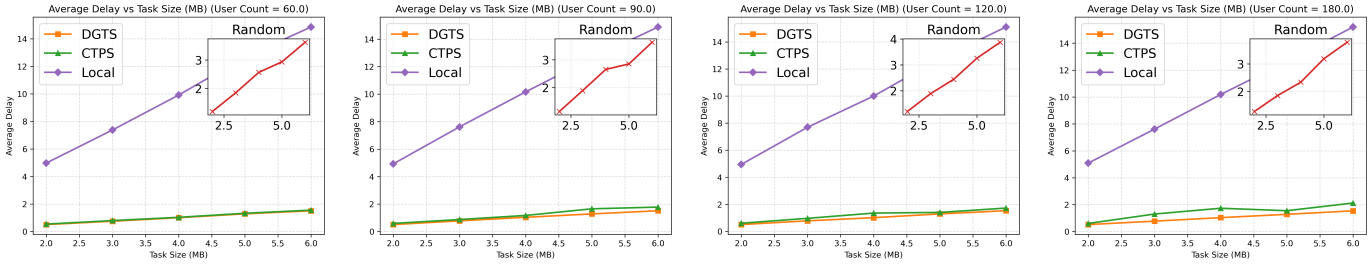

Fig. 3. Average delay versus task size for different UE counts (60, 90, 120, 180) under IoT workloads.

In summary, across all three metrics—cost, delay, and energy—the proposed DGTS achieves numerically identical to that of the solver-based optimal algorithm under the current parameter settings. However, since DGTS relies on local best-response updates instead of exhaustive global search, it does not guarantee global optimality. The equality here indicates that the algorithm happened to reach the global optimum for this instance. In general, DGTS yields near-optimal results while running orders of magnitude faster, making it ideal for latency-sensitive and real-time scheduling where centralized solving is impractical.

C. Large-Scale Results

In the large-scale configuration (UEs: 60–180, Task Size: 2–6 MB), the COCS solver becomes computationally intractable (runtime about one hour for a decision). We therefore compare DGTS against CTPS, Local, and Random. The analysis focuses on the two most critical performance metrics—average cost and average delay—as they jointly capture the overall efficiency and responsiveness of the system. where delay directly impacts QoS.

Figure 2 illustrates the variation of average cost with task size across different UE counts (60–180). DGTS consistently achieves the lowest cost among all algorithms, maintaining near-linear scaling with respect to task size. When the UE count increases, CTPS experiences a sharper cost rise in some scenarios, whereas DGTS dynamically adjusts offloading ratios and transmit power via best-response updates, effectively mitigating interference. On average, DGTS outperforms CTPS by about 10–15%, and achieves over 80% and 99% improvement compared with Local and Random, respectively. The relative gain expands as user density grows, demonstrating DGTS's superior adaptability to heavy-load environments.

Figure 3 presents the corresponding delay trends. DGTS yields the smallest delay across all scenarios, maintaining sub-linear growth even when both UE count and task size increase. For moderate load (e.g., 90–120 UEs), DGTS achieves approximately 12–18% lower delay than CTPS, mainly due to more efficient scheduling of computation tasks between edge and cloud resources. In contrast, Local execution suffers from rapidly growing delay (often exceeding 2–3× that of DGTS) since it lacks collaborative offloading, while Random scheduling leads to frequent bottlenecks in radio access links. As task size increases from 2 MB to 6 MB, the delay gap between DGTS and CTPS further widens, confirming that game-theoretic coordination becomes increasingly beneficial under intensive workloads.

Overall, DGTS matches COCS in small cases and decisively outperforms heuristics at scale, balancing optimality, scalability, and real-time efficiency for large, delay-sensitive NOMA scheduling.

VII. Conclusion

This paper presented a unified hybrid ground–satellite edge computing framework that couples NOMA-based access with multi-layer task offloading and a cross-layer latency–energy objective. The proposed solution framework integrates two complementary components. First, a solver-based optimal scheduling algorithm is developed to obtain the exact global optimum of the formulated MINLP problem, serving as a benchmark for system-level performance under mini-scale experimental settings. Second, a decentralized game-theoretic scheduler is introduced to deliver near-optimal performance with provable convergence and substantially lower computational complexity. Owing to its rapid decision-making capability, this scheduler is particularly suited for latency-sensitive and large-scale deployment scenarios, where centralized global optimization becomes computationally prohibitive. Experiments show that DGTS matches the optimal MINLP in mini-scale tests and, at scale, outperforms CTPS by about

10–15% in cost and 12–18% in delay, with larger gains over Local and Random. The results highlight that coordinating communication, computation, and backhaul is key to practical performance in hybrid ground–satellite systems.

Promising directions include mobility- and queue-aware online control for handovers, ISL congestion, and non-stationary traffic, along with learning-augmented scheduling under partial observability. For dense IoT, prioritize energy-aware wake/sleep, priority slicing, and semantics-aware offloading with privacy under intermittent connectivity.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 62372242, the Natural Science Foundation of Jiangsu Province under Grant BK20240692, and the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant 24KJB520021.

References

[1] L. Zhong, X. Chen, C. Xu, Y. Ma, M. Wang, Y. Zhao, and G.-M. Muntean, "A multi-user cost-efficient crowd-assisted vr content delivery solution in 5g-and-beyond heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 8, pp. 4405–4421, 2022.

[2] J. Wu, Y. Guan, Q. Mao, Y. Cui, Z. Guo, and X. Zhang, "Zgaming: Zero-latency 3d cloud gaming by image prediction," in *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023, pp. 710–723.

[3] Y. Gao, S. Yang, F. Li, S. Trajanovski, P. Zhou, P. Hui, and X. Fu, "Video content placement at the network edge: Centralized and distributed algorithms," *IEEE Transactions on Mobile Computing*, vol. 22, no. 11, pp. 6843–6859, 2022.

[4] Z. Xiao, J. Shu, H. Jiang, J. C. Lui, G. Min, J. Liu, and S. Dustdar, "Multi-objective parallel task offloading and content caching in d2d-aided mec networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 11, pp. 6599–6615, 2022.

[5] G. Zheng, N. Wang, and R. R. Tafazolli, "Sdn in space: a virtual data-plane addressing scheme for supporting leo satellite and terrestrial networks integration," *IEEE/ACM Transactions on networking*, vol. 32, no. 2, pp. 1781–1796, 2023.

[6] S. Yao, Y. Lin, M. Wang, K. Xu, M. Xu, C. Xu, and H. Zhang, "Leoedge: A satellite-ground cooperation platform for the ai inference in large leo constellation," *IEEE Journal on Selected Areas in Communications*, vol. 43, no. 1, pp. 36–50, 2024.

[7] J. Zhou, Q. Yang, L. Zhao, H. Dai, and F. Xiao, "Mobility-aware computation offloading in satellite edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 10, pp. 9135–9149, 2024.

[8] J. Zhou, J. Liang, L. Zhao, S. Wan, H. Cai, and F. Xiao, "Latency-energy efficient task offloading in the satellite network-assisted edge computing via deep reinforcement learning," *IEEE Transactions on Mobile Computing*, 2024.

[9] F. Song, H. Xing, X. Wang, S. Luo, P. Dai, Z. Xiao, and B. Zhao, "Evolutionary multi-objective reinforcement learning based trajectory control and task offloading in uav-assisted mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 12, pp. 7387–7405, 2022.

[10] Y. Liu and H. Cheng, "Outage performance analysis and optimization of two-user downlink noma systems with imperfect sic and improper signaling," *IEEE Transactions on Wireless Communications*, vol. 23, no. 9, pp. 11 649–11 661, 2024.

[11] A. S. De Sena, F. R. M. Lima, D. B. Da Costa, Z. Ding, P. H. Nardelli, U. S. Dias, and C. B. Papadias, "Massive mimo-noma networks with imperfect sic: Design and fairness enhancement," *IEEE Transactions on Wireless Communications*, vol. 19, no. 9, pp. 6100–6115, 2020.

[12] L. Luo, Q. Li, and J. Cheng, "Performance analysis of overlay cognitive noma systems with imperfect successive interference cancellation," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4709–4722, 2020.

[13] C. Tsanikidis and J. Ghaderi, "Scheduling stochastic traffic with end-to-end deadlines in multi-hop wireless networks," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 651–660.

[14] B. Xie, H. Cui, I. W.-H. Ho, Y. He, and M. Guizani, "Computation offloading and resource allocation in leo satellite-terrestrial integrated networks with system state delay," *IEEE Transactions on Mobile Computing*, vol. 24, no. 3, pp. 1372–1385, 2024.

[15] R. Xing, M. Xu, A. Zhou, Q. Li, Y. Zhang, F. Qian, and S. Wang, "Deciphering the enigma of satellite computing with cots devices: Measurement and analysis," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 420–435.

[16] J. Shenoy, O. Chabra, T. Chakraborty, S. Jog, D. Vasisht, and R. Chandra, "Cosmac: Constellation-aware medium access and scheduling for iot satellites," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 724–739.

[17] Y. Ren, A. Gamage, L. Liu, M. Li, S. Chen, Y. Dong, and Z. Cao, "Sateriot: High-performance ground-space networking for rural iot," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 755–769.

[18] Y. Zengshan, W. Changhao, G. Chongbin, L. Yuanchun, X. Mengwei, G. Weiwei, and C. Chuanxiu, "A comprehensive survey of orbital edge computing: Systems, applications, and algorithms," *Chinese Journal of Aeronautics*, vol. 38, no. 7, p. 103316, 2025.

[19] Q. Chen, Z. Guo, W. Meng, S. Han, C. Li, and T. Q. Quek, "A survey on resource management in joint communication and computing-embedded sagin," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 3, pp. 1911–1954, 2024.

[20] V. Singh, T. Chakraborty, S. Jog, O. Chabra, D. Vasisht, and R. Chandra, "Spectrumize: Spectrum-efficient satellite networks for the internet of things," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 825–840.

[21] C. Dong, Y. Tian, Z. Zhou, W. Wen, and X. Chen, "Joint power allocation and task offloading for reliability-aware services in noma-enabled mec," *IEEE Transactions on Wireless Communications*, vol. 23, no. 7, pp. 7537–7551, 2023.

[22] K. M. Kim and T.-J. Lee, "Random access and uplink shared channel resource allocation with noma," *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 6896–6907, 2023.

[23] R. Huang, W. Wen, X. Chen, Z. Zhou, Q. Chen, and C. Dong, "Joint power allocation and task replication for reliability-sensitive services in noma-enabled vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 3, pp. 4178–4193, 2023.

[24] N. Jones and E. Modiano, "Optimal slicing and scheduling with service guarantees in multi-hop wireless networks," in *Proceedings of the Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2024, pp. 181–190.

[25] S. Zhang, S. Dou, Z. Li, K. L. Yeung, and T. Q. Quek, "Oracle: Qos-aware online service provisioning in non-terrestrial networks with safe transfer learning," in *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 2025, pp. 1–10.

[26] W. Liu, J. Wu, Q. Lin, H. Luo, Q. Zhang, K. Qiu, Z. Chen, and Y. Gao, "Efficient satellite-ground interconnection design for low-orbit mega-constellation topology," *IEEE Transactions on Mobile Computing*, 2024.

[27] Q. Tang, R. Xie, Z. Fang, T. Huang, T. Chen, R. Zhang, and F. R. Yu, "Joint service deployment and task scheduling for satellite edge computing: A two-timescale hierarchical approach," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 5, pp. 1063–1079, 2024.

[28] N. Mohan, A. E. Ferguson, H. Cech, R. Bose, P. R. Renatin, M. K. Marina, and J. Ott, "A multifaceted look at starlink performance," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 2723–2734.

[29] L. Zhong, Y. Li, M.-F. Ge, M. Feng, and S. Mao, "Joint task offloading and resource allocation for leo satellite-based mobile edge computing systems with heterogeneous task demands," *IEEE Transactions on Vehicular Technology*, 2025.

[30] F. Minani, M. Kobayashi, T. Fujihashi, M. A. Alim, S. Saruwatari, M. Nishi, and T. Watanabe, "Channel prediction and fair resource allocation for ntn uplinks by lstm and deep reinforcement learning," *IEEE Transactions on Wireless Communications*, 2025.

[31] H. Cheng, M. Zhang, M. Hua, Y. Xia, F. Ding, S. Zhang, W. Pei, and A. L. Swindlehurst, "On the rate region of the downlink noma system with improper signaling and imperfect sic," *IEEE Transactions on Communications*, 2025.

[32] N. Hui, Q. Sun, J. Zeng, L. Tian, Y. Wang, and Y. Zhou, "Mixed numerology-based intelligent resource management in a sliced 6 g

space–terrestrial integrated radio access network," *IEEE Transactions on Mobile Computing*, 2024.

[33] G. Zheng, Q. Ni, K. Navaie, and H. Pervaiz, "Semantic communication in satellite-borne edge cloud network for computation offloading," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 5, pp. 1145–1158, 2024.

[34] D. Li, Y. Sun, J. Peng, S. Cheng, Z. Yin, N. Cheng, J. Liu, Z. Li, and C. Xu, "Dual network computation offloading based on drl for satellite-terrestrial integrated networks," *IEEE Transactions on Mobile Computing*, 2024.