

Resource-Aware Dynamic Scheduling for Energy-Efficient LEO Constellation-as-a-Service

Muhammad Ismail Dad¹, Umber Saleem², Ning Wang², Wei Koong Chai¹

¹School of Computing and Engineering, Bournemouth University, UK

² School of Electrical, Electronic and Mechanical Engineering, University of Bristol, UK

Abstract—Low Earth Orbit (LEO) Constellation-as-a-Service (LEO-CaaS) is emerging as a distributed space-edge computing paradigm. However, satellite mobility makes connectivity and onboard resources time-varying, complicating energy-efficient orchestration for latency-constrained services. We capture these coupled dynamics by formulating joint task scheduling and resource allocation as a mixed-integer nonlinear program that minimizes system energy under latency, visibility, contact-window, communication, and computation constraints. To enable online scheduling, we propose Computation and Resource-Aware Fair Task Scheduling (CRAFTS), a low-complexity two-stage heuristic that combines workload-aware admission, resource-contention-aware satellite ranking, and workload-proportional redistribution. Simulation results under dynamic LEO-CaaS conditions show that CRAFTS reduces total energy by up to 29% compared with a multi-agent learning benchmark while achieving sub-second scheduling. It also consistently lowers energy per assigned task over heuristic baselines, demonstrating scalable online scheduling performance.

Index Terms—Constellation-as-a-Service, LEO, MEC, energy efficiency, latency, scheduling

I. INTRODUCTION

Low Earth Orbit (LEO) constellations are evolving beyond connectivity into Constellation-as-a-Service (CaaS) platforms, driven by advances in space-based networking and the growing demand for global service access [1], [2]. By integrating communication, storage, and onboard computing, LEO-CaaS supports edge services in regions with limited terrestrial infrastructure, enabling applications such as remote sensing analytics, disaster assessment, environmental monitoring, maritime monitoring, and satellite-hosted AI inference [3]. However, sustainable LEO-CaaS operation requires minimizing communication and computation energy while satisfying latency-constrained service delivery over resource-limited satellites. This is challenging due to heterogeneous workloads, strict deadlines, satellite mobility, time-varying user-satellite connectivity, and finite contact duration [4].

Joint computation offloading and resource allocation have been widely studied in terrestrial MEC systems under latency and energy constraints [5]–[7], including mobility-aware offloading [8] and collaborative cross-server scheduling [9]. However, these works generally assume fixed infrastructure and relatively stable connectivity, which do not directly apply to LEO-CaaS systems with intermittent coverage and moving service nodes [1]. Recent LEO-MEC and satellite-

edge studies have considered multi-satellite power/offloading tradeoffs [10], caching and communication resource optimization [11], cooperative satellite computation [12], and joint offloading, bandwidth, computation, and power allocation [13], [14]. Nevertheless, many rely on convexification, relaxed association variables, or simplified dynamics, limiting their scalability under rapidly varying user-satellite visibility.

Learning-based methods have also been explored to improve adaptability under dynamic LEO conditions. Yang et al. [15] employed a parameterized deep Q-network for joint offloading and resource allocation, while multi-agent reinforcement learning has been proposed for distributed satellite computation offloading [16]. More closely, Hassan et al. [17] proposed an attention-based cooperative multi-agent PPO (Co-MAPPO) framework for satellite-based data offloading and computation. Although learning-based methods can adapt to nonlinear and time-varying conditions after training, they require substantial training effort and may face convergence, feasibility, and scalability challenges in large LEO-CaaS systems.

Motivated by the limited support for lightweight decision-making under dynamic user-satellite association and coupled onboard capacity limits, we investigate energy-efficient service orchestration for LEO-CaaS from a joint assignment-and-allocation perspective. In this setting, selecting a serving satellite simultaneously shapes the communication opportunity, available onboard processing capacity, and service time window for each request. We therefore propose Computation- and Resource-Aware Fair Task Scheduling (CRAFTS), an online heuristic that exploits spatio-temporal connectivity and capacity-aware indicators to guide feasible low-energy service decisions.

The main contributions of this paper are summarized as follows:

- We formulate a time-coupled mixed-integer nonlinear program (MINLP) for sustainable LEO-CaaS service delivery, jointly optimizing user-satellite assignment, bandwidth allocation, and onboard computation allocation under latency, connectivity, contact-duration, and resource-capacity constraints.
- We develop CRAFTS as an online, low-complexity service orchestration framework that combines workload-aware task prioritization, contact-aware satellite rank-

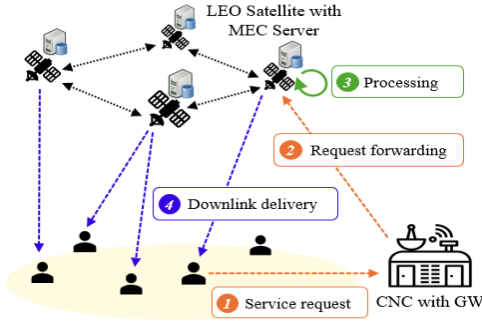


Fig. 1. System model of the proposed LEO-CaaS framework illustrating end-to-end service delivery.

ing, and contention-aware resource reservation to support feasible and energy-efficient task execution.

- Through dynamic LEO-CaaS simulations, we show that CRAFTS achieves lower energy per assigned task, sub-second online scheduling, and substantially lower run-time than a multi-agent learning benchmark, while consistently outperforming representative heuristic baselines.

II. SYSTEM MODEL

A. Network Model

We consider a multi-user, multi-server LEO-CaaS framework that provides edge computing services in remote or disaster-affected areas with limited terrestrial infrastructure [1]. Representative applications include remote sensing analytics, environmental monitoring, disaster assessment, and AI inference on satellite-hosted datasets. The system consists of a centralized network controller (CNC), a set of users $\mathcal{U} = \{1, 2, \dots, U\}$, and a set of LEO satellites $\mathcal{L} = \{1, 2, \dots, L\}$, as shown in Fig. 1. Users submit computation requests to the CNC through a local ad-hoc Wi-Fi network. The CNC maintains global network state information and performs scheduling and resource allocation, after which selected satellites execute the tasks using onboard edge servers and deliver the processed results through downlink service links.

B. LEO Satellites and User Association

We consider the mobility of LEO satellites over a time horizon T . To manage the dynamics of satellite movement, we discretize the time into N slots of equal duration τ , such that $T = N\tau$. The set of slots is denoted as $\mathcal{N} = \{1, 2, \dots, N\}$. The slot duration is chosen such that satellite positions remain quasi-static within each slot. The satellites' mobility is assumed known in advance, as LEO ephemeris is deterministic and can be accurately predicted from orbital data.

Based on the LEO satellite-user geometry, the slant range between user i and satellite j in slot n is $d_{ij}^n = \sqrt{(R_e + h_j^n)^2 - R_e^2 \cos^2(\theta_{ij}^n)} - R_e \sin(\theta_{ij}^n)$, where R_e is the

Earth radius, h_j^n is the satellite altitude, and θ_{ij}^n is the elevation angle.

Due to the satellites' motion, users may move in and out of coverage over time T . This is captured by the binary visibility variable $x_{ij}^n \in \{0, 1\}$, which indicates whether user i is within the coverage of satellite j in slot n . The variable is determined from the elevation angle θ_{ij}^n as follows:

$$x_{ij}^n = \begin{cases} 1, & \text{if } \theta_{ij}^n > \theta_{\min} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

where $\theta_{\min}$ is the minimum required elevation angle for effective communication between user and satellite in a slot.

Building on the visibility matrix, we define the assignment variable $a_{ij}^n \in \{0, 1\}$, where $a_{ij}^n = 1$ indicates that user i is served by satellite j in slot n . Assignment is allowed only under visibility and each user can be associated with at most one satellite per slot, while each satellite may serve multiple users subject to its communication and computation resources.

To capture task-level service continuity, we define the remaining contact duration between user i and satellite j from slot n as $T_{ij}^{n,r} = \tau L_{ij}^{n,r}$, where $L_{ij}^{n,r}$ is the number of consecutive future slots from slot n for which the visibility condition remains satisfied. This value is obtained from the predicted user-satellite visibility sequence.

C. Communication Model

We assume that bandwidth is orthogonally shared using OFDMA among different LEO satellites and the assigned users. Let B_j denote the total available bandwidth for each satellite, and b_{ij}^n denote the bandwidth allocated to user i by satellite j in slot n . Then, the achievable data rate is given as

$$r_{ij}^n = b_{ij}^n \log_2 \left(1 + \frac{p_{ij}^n |h_{ij}^n|^2}{\sigma_o} \right), \quad (2)$$

where p_{ij}^n denotes power assigned for user i by satellite j in slot n . σ_o represents the additive white Gaussian noise power. We assume that, power allocation is fixed for all assigned users. h_{ij}^n represents the channel power gain between user i and LEO satellite j in slot n and is provided as

$$h_{ij}^n = \eta_{ij}^n \Gamma_{ij}^n d_{ij}^{n-\zeta}, \quad (3)$$

where η_{ij}^n is a complex Gaussian variable for Rayleigh fading, Γ_{ij}^n is for shadow fading, and ζ is the path loss exponent. d_{ij}^n is the slant range between LEO satellite j and user i during slot n which is already defined in Section II-B.

III. LEO-EDGE COMPUTING AND PROBLEM FORMULATION

Each user requests a computation service from the LEO-CaaS system. We assume that the data, content, or models required to serve user requests are available onboard the satellites [18]. The selected LEO satellite executes the requested task using onboard computing resources and locally

available data or models, after which the processed output is delivered to the user through the downlink service link. Each task is represented by $Q_i^n = \{D_i^n, C_i^n, \delta_i^n\}$, where D_i^n denotes the input data size, C_i^n the required computation per bit, and δ_i^n the delay threshold. We avoid mid-task handover by enforcing contact-feasible task completion. Thus, satellite reassociation can occur only between consecutive task requests, not during the execution of an admitted task.

Accordingly, the total service time is modeled as the sum of onboard computation and downlink transmission delays. Let f_{ij}^n denote the computation resource assigned by LEO j to task i during slot n . Then computation delay is given as

$$T_{ij}^{n,c} = \frac{D_i^n C_i^n}{f_{ij}^n}. \quad (4)$$

Based on the communication model in Section II-C, the downlink transmission delay for delivering the processed output from LEO satellite j to user i in slot n is given by

$$T_{ij}^{n,t} = \frac{\rho_i D_i^n}{r_{ij}^n}, \quad (5)$$

where $\rho_i \in (0, 1]$ denotes the output-to-input data ratio of task i , such that the processed output size is $D_{i,\text{out}}^n = \rho_i D_i^n$.

Let P_j denote the energy dissipation per unit time for LEO satellite j . Then, the energy dissipated by LEO j for serving request of user i in slot n is derived in terms of processing and transmission delays as

$$E_{ij}^n = a_{ij}^n (P_j T_{ij}^{n,c} + P_j T_{ij}^{n,t}). \quad (6)$$

Let $\mathbf{a} = \{a_{ij}^n\}$, $\mathbf{b} = \{b_{ij}^n\}$, and $\mathbf{f} = \{f_{ij}^n\}$ collect the assignment, bandwidth, and computation-allocation variables over all $i \in \mathcal{U}$, $j \in \mathcal{L}$, and $n \in \mathcal{N}$. Based on the per-service-request energy in (6), we formulate the joint task scheduling and resource allocation problem for system-wide energy minimization over the considered time horizon as follows:

$$\mathcal{P}_0 : \min_{\mathbf{a}, \mathbf{b}, \mathbf{f}} \sum_{j=1}^L \sum_{n=1}^N \sum_{i=1}^U E_{ij}^n \quad (7a)$$

$$\text{s.t.} \sum_{j=1}^L a_{ij}^n \leq 1, \forall i \in \mathcal{U}, \forall n \in \mathcal{N} \quad (7b)$$

$$a_{ij}^n \leq x_{ij}^n, \forall i \in \mathcal{U}, \forall j \in \mathcal{L}, \forall n \in \mathcal{N} \quad (7c)$$

$$\sum_{i=1}^U a_{ij}^n b_{ij}^n \leq B_j, \forall j \in \mathcal{L}, \forall n \in \mathcal{N} \quad (7d)$$

$$\sum_{i=1}^U a_{ij}^n f_{ij}^n \leq F_j, \forall j \in \mathcal{L}, \forall n \in \mathcal{N} \quad (7e)$$

$$\sum_{j=1}^L a_{ij}^n (T_{ij}^{n,c} + T_{ij}^{n,t}) \leq \min\{\delta_i^n, \Delta_i^{n,r}\}, \forall i, n \quad (7f)$$

$$a_{ij}^n \in \{0, 1\}, \forall i \in \mathcal{U}, \forall j \in \mathcal{L}, \forall n \in \mathcal{N} \quad (7g)$$

$$0 \leq b_{ij}^n \leq B_j, \forall i \in \mathcal{U}, \forall j \in \mathcal{L}, \forall n \in \mathcal{N} \quad (7h)$$

$$0 \leq f_{ij}^n \leq F_j, \forall i \in \mathcal{U}, \forall j \in \mathcal{L}, \forall n \in \mathcal{N} \quad (7i)$$

Constraint (7b) allows each user request to be assigned to at most one LEO satellite per slot, while Constraint (7c) permits assignment only to visible satellites. Constraints (7d) and (7e) enforce the per-satellite bandwidth and computation-capacity limits, respectively. Constraint (7f) ensures that each admitted task is completed within both its latency deadline and the remaining contact duration of the selected satellite, where $\Delta_i^{n,r} = \sum_{j=1}^L a_{ij}^n T_{ij}^{n,r}$, thereby avoiding mid-task handover. Constraints (7g)–(7i) define the variable domains and bounds.

The resulting problem is a time-coupled MINLP with binary task-assignment variables a_{ij}^n and continuous resource-allocation variables b_{ij}^n and f_{ij}^n . The coupling between assignment, bandwidth allocation, computation allocation, and latency/contact-time constraints makes direct optimization computationally challenging.

Theorem 1: The joint task scheduling and resource allocation problem $\mathcal{P}_0$ is NP-hard.

Proof: Consider a restricted instance with one time slot, fixed visibility, and fixed resource allocation for each feasible user-satellite association. The remaining decision is to assign tasks to satellites under bandwidth, computation, and latency constraints. This is equivalent to the generalized assignment problem, where tasks are items and satellites are knapsacks with limited resources. With a single satellite, it further reduces to the 0/1 knapsack problem, which is NP-hard. Since $\mathcal{P}_0$ generalizes this restricted case by including multiple slots and continuous resource allocation, $\mathcal{P}_0$ is NP-hard [19].

IV. PROPOSED APPROACH

We present CRAFTS, a centralized resource-management scheme executed at the CNC for joint task assignment and onboard resource allocation in LEO-CaaS. Since solving the MINLP in real time is impractical, CRAFTS uses global knowledge of user workloads, satellite visibility, channel conditions, and residual onboard resources to make low-complexity energy-aware scheduling decisions. It consists of two stages: workload-prioritized user-satellite assignment and proportional residual-resource redistribution.

In each slot, users are processed in descending order of computation workload $D_i^n C_i^n$, so that resource-demanding tasks are considered earlier. For each user, CRAFTS evaluates only visible satellites. Unlike rate-centric association, which may concentrate tasks on favorable-link satellites and cause bandwidth/computation congestion, CRAFTS ranks candidate satellites using both service quality and residual resource availability. Specifically, each visible satellite is scored as $\Psi_{ij}^n = \alpha \tilde{r}_{ij}^n - \beta (\tilde{b}_j^n + \tilde{f}_j^n)$, where $\tilde{r}_{ij}^n$ is the normalized transmission rate, and $\tilde{b}_j^n$ and $\tilde{f}_j^n$ are the normalized fractions of bandwidth and computation resources already allocated on satellite j in slot n . The coefficients α and β control the tradeoff between link quality and current satellite load; both are set to 0.5 to assign equal weight in the simulations.

Algorithm 1 CRAFTS

```

1: Input:  $U, L, N, D_i^n, C_i^n, B_j, F_j, \mathbf{x}, r_{ij}^n, T_{ij}^{n,r}, \delta_i^n$ 
2: Init:  $a_{ij}^n = 0, b_{ij}^n = 0, f_{ij}^n = 0, \alpha, \beta$ 
3: for  $n = 1$  to  $N$  do
4:   Sort users by workload  $D_i^n C_i^n$  in descending order
5:   Initialize remaining resources  $B_j^{rem}, F_j^{rem}$ 
6:   Step 1: Feasible assignment
7:   for each user  $i$  do
8:     Determine visible set  $\mathcal{V}_i^n = \{j | x_{ij}^n = 1\}$ 
9:     Rank satellites  $j \in \mathcal{V}_i^n$  using utility  $\Psi_{ij}^n$ 
10:    for each ranked satellite  $j' \in \mathcal{V}_i^n$  do
11:      Compute feasible  $b_{ij'}^{n,min}, f_{ij'}^{n,min}$  satisfying (7f)
12:      if  $b_{ij'}^{n,min} \leq B_{j'}^{rem}$  and  $f_{ij'}^{n,min} \leq F_{j'}^{rem}$  then
13:        Assign  $a_{ij'}^n = 1$  and reserve  $b_{ij'}^{n,min}, f_{ij'}^{n,min}$ 
14:        Update  $B_{j'}^{rem}, F_{j'}^{rem}$  and break
15:      end if
16:    end for
17:  end for
18:  Step 2: Proportional update
19:  for each satellite  $j$  do
20:    Redistribute  $B_j^{rem}, F_j^{rem}$  over assigned users  $\mathcal{U}_j^n$ 
    proportional to  $D_i^n C_i^n$ 
21:  end for
22: end for
23: Output:  $\mathbf{a}^*, \mathbf{b}^*, \mathbf{f}^*$ 

```

After ranking, CRAFTS checks whether each candidate can complete the task within the effective service time determined by the task deadline and remaining contact duration. If feasible, the task is assigned and the minimum required bandwidth and computation resources are reserved. Finally, residual bandwidth and computation capacity on each satellite are redistributed among assigned users in proportion to $D_i^n C_i^n$, improving resource utilization and supporting larger workloads.

From a computational perspective, the dominant cost of CRAFTS comes from per-slot user sorting and satellite ranking. The assignment stage has complexity $\mathcal{O}(N(U \log U + UL \log L))$, due to per-slot user sorting and per-user satellite ranking. The resource redistribution stage adds $\mathcal{O}(NU)$ over all admitted users. Therefore, the overall complexity is dominated by the assignment stage and is $\mathcal{O}(NUL \log L)$.

V. PERFORMANCE EVALUATION

For simulations, we use MATLAB's Satellite Communications Toolbox to generate a Walker-Star LEO constellation with 60 satellites in 6 orbital planes, following an Iridium-like polar configuration with altitude 781 km and inclination 86° . Satellite mobility and user visibility are evaluated over a 30-min horizon, with users randomly distributed within a 1000 km service region. The time-varying visibility matrix is generated from satellite positions, user locations, and a minimum elevation angle of $\theta^{\min} = 15^\circ$. Key parameters are summarized in Table I.

TABLE I
SIMULATION PARAMETERS

Symbol	Value	Symbol	Value
U	[50, 100]	L	60
T	30 min	τ	1 s
N	1800 slots	P_j	5 W
D_i^n	[100, 500] Kb	C_i^n	[1000, 1500] cycles/bit
δ_i^n	[50, 1000] ms	p_{ij}^n	37 dBm
F_j	[20, 25] GHz	B_j	[5, 10] MHz

CRAFTS is compared with four benchmarks: (i) **Greedy**, which selects the feasible visible satellite with the highest rate; (ii) **Random**, which selects a feasible visible satellite uniformly at random; (iii) **Uniform**, which uses round-robin association and equal resource distribution; and (iv) **Co-MAPPO**, an attention-based multi-agent learning baseline adapted from [17]. Heuristic results are averaged over 20 independent runs. Due to its high training and inference cost, Co-MAPPO is evaluated in a reduced-scale setting with 200 training episodes and 20 test episodes. The evaluation metrics are total energy consumption, corresponding to $\mathcal{P}_0$ in (7a), and average energy per assigned task to account for different task admission rates.

To benchmark against a learning-based scheduler, Fig. 2 compares CRAFTS with Co-MAPPO under a reduced-scale LEO-CaaS scenario, where U varies from 10 to 50, with $L = 40$ and $T = 6$ min. This experiment assesses whether a trained multi-agent policy can provide competitive scheduling decisions as the number of users increases. The results show that Co-MAPPO can learn feasible assignments; however, under the considered training budget and dynamic LEO-CaaS setting, it does not outperform CRAFTS and can even consume more energy than the Greedy baseline, while incurring substantially higher computational overhead. At $U = 50$, Co-MAPPO drops more tasks, which can reduce total energy despite degraded service performance.

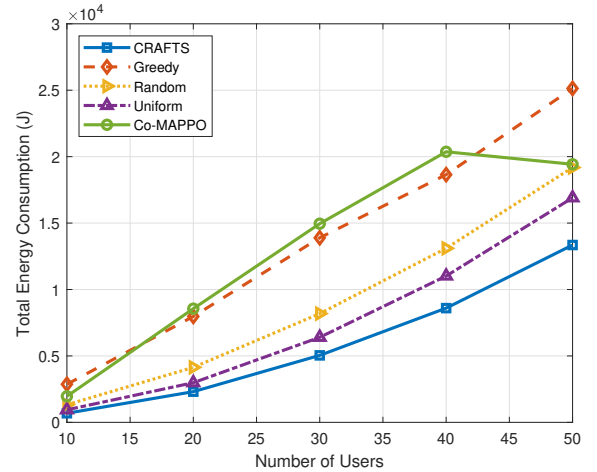


Fig. 2. Reduced-scale LEO-CaaS comparison with Co-MAPPO.

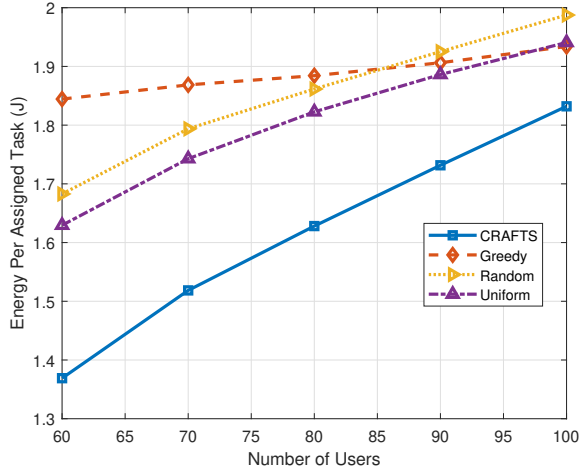


Fig. 3. Energy consumption per assigned task under increasing user density.

To further quantify scalability, Table II reports the average online decision time of CRAFTS, along with Co-MAPPO inference and offline training times. CRAFTS maintains sub-second runtime as U increases from 10 to 50, whereas Co-MAPPO inference grows by about 5.1 times and remains substantially higher. At $U = 50$, CRAFTS completes scheduling in 0.117 s, compared with 42.24 s for Co-MAPPO inference, corresponding to approximately 360 times lower online runtime.

This online-runtime gap is critical in LEO-CaaS, where mobility-driven changes in visibility, channel quality, and contact duration require frequent scheduling updates. Slow inference may produce stale association and resource-allocation decisions, while CRAFTS enables timely sub-second scheduling. Moreover, Fig. 2 shows that CRAFTS reduces total energy by approximately 29% at $U = 50$, demonstrating a better energy-complexity tradeoff without offline training.

We further assess scalability beyond the reduced Co-MAPPO setting. Fig. 3 evaluates a larger LEO-CaaS scenario with up to 100 users, $L = 40$ satellites, and a $T = 30$ min horizon. This experiment measures energy efficiency under growing resource contention. CRAFTS consistently achieves the lowest energy per assigned task by jointly exploiting link quality and residual satellite resources during association. At $U = 100$, CRAFTS provides a consistent 5–8% reduction in energy per assigned task, indicating that it serves admitted tasks more energy-efficiently rather than relying

TABLE II
RUNTIME COMPARISON IN SECONDS UNDER VARYING USER DENSITY.

Scheme	$U = 10$	$U = 30$	$U = 50$
CRAFTS	0.039	0.074	0.117
Co-MAPPO inference	8.36	11.40	42.24
Co-MAPPO training	2617.9	11271	11748

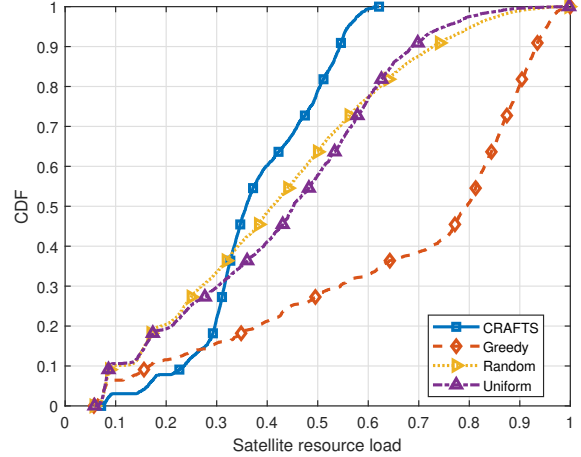


Fig. 4. CDF of normalized satellite resource load.

on task rejection. The gap narrows at high user density as bandwidth and computation constraints become active for most candidate assignments.

In order to understand the source of CRAFTS' energy gain, Fig. 4 plots the CDF of normalized resource usage across active satellite-slot samples for $U = 60$ and $L = 40$. The x-axis represents the normalized load of one active satellite in one slot, with larger values indicating higher bandwidth and computation utilization. CRAFTS shifts the distribution away from high-load regions compared with Greedy, Random, and Uniform, showing that it avoids concentrating tasks on a small subset of satellites. This resource-balanced behavior preserves residual capacity and contributes to lower energy per assigned task under dense deployments.

Next, we evaluate robustness under heavier service demands. Fig. 5 reports the average energy per assigned task as task size increases. Energy rises for all schemes due to higher transmission and computation requirements, but CRAFTS consistently remains the most efficient by prioritizing high-workload tasks and avoiding resource-congested satellites. At 500 Kb, it reduces energy by approximately 36%, 26%, and 22% compared with Greedy, Random, and Uniform, respectively.

Fig. 6 further evaluates scalability with increasing satellite availability. Larger L reduces energy by providing more feasible association and resource-selection options, with CRAFTS benefiting most because its ranking jointly considers link quality and residual resource occupancy. At $L = 60$, CRAFTS reduces energy by approximately 37%, 27%, and 22% compared with Greedy, Random, and Uniform, respectively. The gain gradually saturates as the bottleneck shifts from visibility diversity to task demand and per-satellite resource constraints.

The results show that CRAFTS is scalable and practical for dynamic LEO-CaaS orchestration. Its sub-second deci-

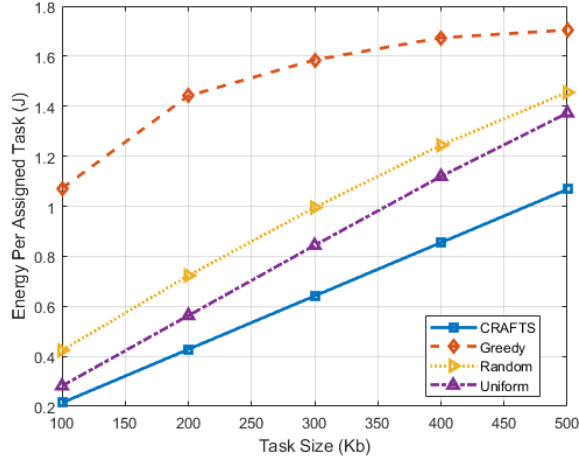


Fig. 5. Energy consumption per assigned task under varying task sizes.

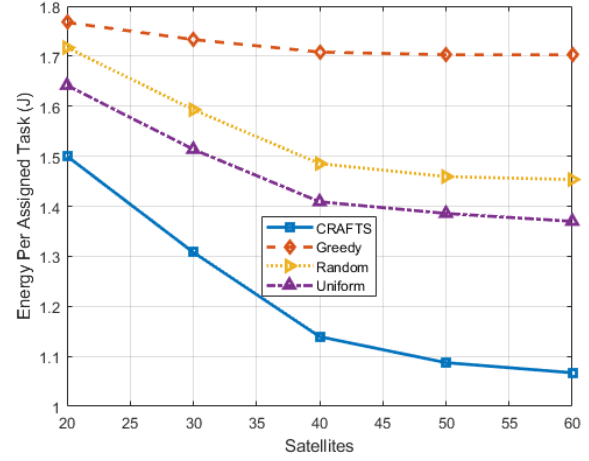


Fig. 6. Energy consumption per assigned task under varying satellites.

sion time supports online operation under changing visibility and resource availability, while consistent energy-per-task gains across user density, task size, and satellite availability indicate robustness to workload and network-scale variations.

VI. CONCLUSION

This paper investigated energy-efficient service orchestration for dynamic LEO-CaaS systems to support sustainable service delivery over satellite constellations acting as distributed service platforms. A joint task scheduling and resource allocation problem was formulated to capture the coupling among visibility, finite contact duration, latency, and onboard communication/computation resources. The results showed that link-aware association alone is insufficient, as workload heterogeneity and satellite resource contention also strongly affect scheduling efficiency. The proposed CRAFTS addressed these factors through contact-aware ranking, contention-aware resource reservation, and workload-proportional redistribution. The evaluation demonstrated a favorable energy-complexity tradeoff, supporting scalable online scheduling for dynamic LEO-CaaS environments.

REFERENCES

- [1] Z. Yin, C. Wu, C. Guo, Y. Li, M. Xu, W. Gao, and C. Chi, "A comprehensive survey of orbital edge computing: Systems, applications, and algorithms," *Chin. J. Aeronaut.*, vol. 38, no. 7, p. 103316, 2025.
- [2] J. A. Fraire, O. Iova, and F. Valois, "Space-terrestrial integrated internet of things: Challenges and opportunities," *IEEE Commun. Mag.*, vol. 60, no. 12, pp. 64–70, 2022.
- [3] J. Liu, "Overview of Low Earth Orbit satellite communication systems," in *Proc. 3rd Int. Conf. Softw. Eng. Mach. Learn.*, 2025.
- [4] A. Li, T. Zhou, T. Xu, Y. Ouyang, H. Hu, and C. Wu, "Leo satellite assisted edge computing with latency and energy optimization," *IEEE Trans. Netw. Sci. Eng.*, 2025.
- [5] Y. Sun and Q. He, "Joint task offloading and resource allocation for multi-user and multi-server mec networks: A deep reinforcement learning approach with multi-branch architecture," *Eng. Appl. Artif. Intell.*, vol. 126, p. 106790, 2023.
- [6] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, p. 1446, 2019.
- [7] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, 2019.
- [8] S. Pang, M. Zhou, H. Gui, X. He, N. Wang, and L. Wang, "Load-balanced multi-user mobility-aware task offloading in multi-access edge computing," *Comput. Commun.*, vol. 235, p. 108102, 2025.
- [9] Y. Shi, Y. Xia, and Y. Gao, "Cross-server computation offloading for multi-task mobile edge computing," *Info*, vol. 11, no. 2, p. 96, 2020.
- [10] Y. Li, S. Zhu, T. Xiong, Y. Li, Q. Su, and J. Dai, "Joint task offloading and power allocation for satellite edge computing networks," *Sensors*, vol. 25, no. 9, p. 2892, 2025.
- [11] Y. Hao, Z. Song, Z. Zheng, Q. Zhang, and Z. Miao, "Joint communication, computing, and caching resource allocation in leo satellite mec networks," *IEEE Access*, vol. 11, pp. 6708–6716, 2023.
- [12] Z. Song, Y. Hao, and X. Sun, "Computation offloading and resource allocation algorithm for collaborative leo satellite multi-access edge computing," *Acta Electron. Sin.*, vol. 50, no. 3, pp. 567–573, 2022.
- [13] M. Wang, Y. Wang, C. Zhang, H. Zhou, and L. Yi, "Joint computation offloading and resource allocation for low-earth orbit satellites mec networks," in *International Conference on Wireless and Satellite Systems*. Springer, 2024, pp. 150–160.
- [14] K. Wei, Q. Tang, J. Guo, M. Zeng, Z. Fei, and Q. Cui, "Resource scheduling and offloading strategy based on leo satellite edge computing," in *IEEE 94th Veh. Technol. Conf.*, 2021, pp. 1–6.
- [15] X. Yang, H. Fang, Y. Gao, X. Wang, K. Wang, and Z. Liu, "Computation offloading and resource allocation based on p-dqn in leo satellite edge networks," *Sensors*, vol. 23, no. 24, p. 9885, 2023.
- [16] Q. Xiu, J. Liu, X. Liu, and J. Wang, "Computation offloading and resource allocation in satellite edge computing networks: A multi-agent reinforcement learning approach," *Comput. Netw.*, p. 111680, 2025.
- [17] S. S. Hassan, Y. M. Park, Y. K. Tun, W. Saad, Z. Han, and C. S. Hong, "Satellite-based its data offloading & computation in 6g networks: A cooperative multi-agent proximal policy optimization drl with attention approach," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4956–4974, 2024.
- [18] B. Chintalapati, "Opportunities and challenges of on-board ai-based image recognition in small satellite missions," *Acta Astronaut.*, 2025.
- [19] D. Pisinger, "Where are the hard knapsack problems?" *Comput. Oper. Res.*, vol. 32, no. 9, pp. 2271–2284, 2005.