Adaptive Drone-Assisted IoT Communications: An RL Framework for Optimal Base-Station Placement and Traffic Load Balancing

Abee Alazzwi*, Petro M. Tshakwanda*, Henok B. Tsegaye*, Michael Devetsikiotis*, Harsh Kumar†
*Dept. of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, USA
Email: {abeer88, pmushidi, htsegaye, mdevets}@unm.edu
†R. B. Annis School of Engineering, University of Indianapolis, Indianapolis, IN, USA
Email: kumarh@uindy.edu

***Abstract*—We present a Hierarchical Actor-Critic (HAC) reinforcement-learning framework for adaptive drone-assisted Internet of Things (IoT) communications. The framework jointly optimizes Drone Base Station (DBS) placement and user association under realistic wireless backhaul capacity and M/M/1 queueing-delay constraints. By separating slow-timescale DBS positioning from fast user association and coordinating both through a shared global reward, HAC enables scalable, low-latency, and fairness-aware control in dense IoT networks. Extensive simulations over a 1000×1000 m macrocell show that HAC reduces average end-to-end delay by roughly an order of magnitude compared to a greedy SINR-based policy and slightly outperforms a flat deep RL baseline, while improving load-balancing fairness and maintaining queue stability under heavy traffic.**

***Index Terms*—UAV communications, drone base station (DBS), Internet of Things (IoT), reinforcement learning, hierarchical actor–critic, queueing delay, traffic load balancing.**

I. Introduction

Drone-assisted wireless networks are a flexible way to boost coverage and capacity in massive IoT deployments, complementing terrestrial cellular infrastructure in hard-to-reach or time-varying environments such as disaster response, temporary events, and rural areas [1]–[3]. Unmanned Aerial Vehicles (UAVs) acting as drone base stations (DBSs) can be rapidly deployed to offload congested Macro Base Stations (MBSs), extend coverage, and adapt to spatially heterogeneous traffic.

However, the joint optimization of DBS placement and user association in IoT networks is challenging. Each DBS must decide *where* to position itself in 3-D space and *which* users to serve, under coupled constraints imposed by the wireless backhaul, access bandwidth, and queue stability. Static or coverage-driven placement strategies often ignore backhaul saturation and queueing dynamics, leading to overloaded links and unstable delays. Moreover, heuristic user association rules may produce an unfair load distribution and degrade the quality of service for edge users. Similar load-balancing and backhaul-awareness issues appear in heterogeneous cellular networks, where inappropriate user association can overload small cells [4], [5].These interactions create a joint optimization problem over DBS positions and user association that is difficult to solve optimally in real time.

Deep reinforcement learning (DRL) has recently been applied to UAV trajectory control and resource allocation in wireless networks [6]–[8]. Nevertheless, many existing approaches treat placement and user association separately, neglect queueing-theoretic delay, or assume ideal backhaul. Furthermore, single-agent ("flat") RL policies that jointly control placement and association suffer from large state–action spaces and slow convergence in dense IoT scenarios.

In this work, we address these challenges by introducing a hierarchical actor–critic (HAC) framework that jointly optimizes DBS placement and user association under realistic access–backhaul coupling and queueing delay. A high-level agent updates DBS positions on a slow timescale based on long-term latency and fairness, while a low-level agent performs fast user association decisions based on instantaneous channel and queue states. Both agents are coordinated through a shared reward that internalizes backhaul constraints, M/M/1 access delay, and fairness.

The main contributions of this paper are:

- We formulate the joint DBS placement and user association problem in a drone-assisted IoT network with explicit modeling of air-to-ground channels, wireless backhaul links, and M/M/1 access queues, targeting average end-to-end latency and load-balancing fairness under stability constraints.
- We design a hierarchical actor-critic (HAC) framework that decouples long-term DBS placement from short-term user association while coordinating both via a shared reward that embeds delay, backhaul congestion, and Jain's fairness index.
- We provide an explicit training algorithm, complexity and scalability analysis with respect to the numbers of users and DBSs, and an ablation study contrasting hierarchical versus flat RL as well as variants with decoupled rewards.

TABLE I
COMPARISON WITH RECENT DRL/MARL-BASED UAV NETWORKING WORKS

Work	Backhaul	Queueing	Hierarchy	Fairness
Alablani *et al.* [9]	✓	–	–	–
Dai *et al.* [8]	–	–	–	–
Cheng *et al.* [10]	✓	–	MARL	–
Chen *et al.* [11]	–	–	✓	–
Kim *et al.* [12]	–	–	✓	✓
This work (HAC)	✓	✓	✓	✓

- Through simulations in a dense IoT macrocell, we show that HAC significantly reduces delay relative to a greedy SINR-based baseline and improves or matches fairness compared to both greedy and flat RL, while maintaining stable queues under heavy traffic.

A. Recent Advances and Positioning of This Work

Beyond the foundational UAV and mobile edge computing literature [1]–[3], several recent works have applied DRL and multi-agent RL (MARL) to UAV-assisted networks. Dai *et al.* use MARL to jointly optimize user association and 3-D trajectories in full-duplex multi-UAV networks, demonstrating significant gains in throughput and coverage [8]. Qin *et al.* consider deep RL-based resource allocation and trajectory planning in integrated sensing and communication (ISAC) UAV networks, showing that learned policies can outperform heuristic baselines under joint sensing-communications constraints [7]. Feriani and Hossain survey single- and multi-agent deep RL techniques for AI-enabled wireless networks, highlighting open challenges in scalability, stability, and constraint handling [6].

On the user-association side, Alablani *et al.* propose a DQN–GNN scheme that combines deep Q-learning with graph neural networks to capture spatial correlations between users and base stations, improving association decisions compared to conventional policies [9]. Cheng *et al.* design a learning-based user association and dynamic resource allocation framework for multi-connectivity UAV networks, where MARL coordinates multiple agents under QoS and backhaul constraints [10]. Hierarchical RL ideas have also been explored for UAV and ISAC systems. Chen *et al.* develop a hierarchical DRL approach for throughput maximization in reconfigurable intelligent surface-aided UAV ISAC networks [11], while Kim *et al.* introduce HiMAQ, a hierarchical multi-agent Q-learning framework to simultaneously improve throughput and fairness in UAV-aided IoT networks [12].

In parallel, HetNet works such as [4], [5] show that user association must carefully account for backhaul limitations and traffic differentiation to avoid overloading small cells. However, most DRL and MARL approaches mentioned above either assume ideal backhaul, omit queueing-theoretic latency, or focus on trajectory control without explicitly linking access and backhaul layers.

Compared to these recent studies, the proposed HAC framework targets a different operating point. First, it explicitly integrates wireless backhaul limitations and queueing-aware M/M/1 access delay into the RL reward, thereby internalizing access–backhaul coupling and queue stability. Second, it uses a bi-level hierarchical design aligned with natural network timescales (slow DBS placement vs. fast user association), while coordinating both levels through a shared global reward that also enforces Jain's fairness index. Third, HAC is evaluated under dense IoT traffic with heavy loads, demonstrating latency and fairness gains over coverage-driven, greedy load balancing [5], and flat RL baselines. This combination of backhaul-aware, queueing-aware, fairness-aware, and hierarchical control is, to the best of our knowledge, not jointly addressed in prior DRL/MARL-based UAV networking works.

II. SYSTEM MODEL

We consider a drone-assisted IoT network comprising one macro base station (MBS), a set of DBSs $\mathcal{D}$, and a set of IoT users $\mathcal{U}$ randomly distributed over a macrocell of area $1000\,\mathrm{m} \times 1000\,\mathrm{m}$. Time is slotted. Each DBS $d \in \mathcal{D}$ is located at 3-D coordinates $\mathbf{p}_d = (x_d, y_d, h_d)$ and is connected to the MBS via a wireless backhaul link of capacity $C_d^{(b)}$ (in $\mathrm{bit\,s^{-1}}$). Users can associate either with the MBS or with one DBS depending on channel conditions and load. We denote by $\alpha_{ud} \in \{0, 1\}$ a binary association variable indicating whether user u is served by DBS d; the MBS association is handled by a similar indicator. The resulting joint placement and association problem resembles load-balancing in HetNets [4], [5], but with dynamic UAV positions and wireless backhaul links.

A. Air-to-Ground Channel Model

The air-to-ground link between DBS d and user u follows an empirical line-of-sight (LoS) probability model, as in [13]. Let θ_{ud} denote the elevation angle of DBS d as seen from user u. The LoS probability is

$$P_{\mathrm{LoS}}(\theta_{ud}) = \frac{1}{1 + a\exp\big(-b(\theta_{ud} - a)\big)}, \tag{1}$$

where a, b are environment-dependent constants. The average path loss is

$$L_{ud} = P_{\mathrm{LoS}} L_{ud}^{\mathrm{LoS}} + (1 - P_{\mathrm{LoS}}) L_{ud}^{\mathrm{NLoS}}, \tag{2}$$

with L_{ud}^{LoS} and L_{ud}^{NLoS} denoting the deterministic LoS and NLoS path losses, respectively.

The received power at user u from DBS d is

$$P_{ud}^{\mathrm{rx}} = P_d^{\mathrm{tx}} - L_{ud}, \tag{3}$$

and the corresponding signal-to-interference-plus-noise ratio (SINR) is

$$\gamma_{ud} = \frac{P_{ud}^{\mathrm{rx}}}{I_u + N_0 B}, \tag{4}$$

where I_u is the aggregate interference, N_0 is the noise spectral density, and B is the access bandwidth.

B. Access and Backhaul Rates

The achievable access rate for user u when associated with BS $s(u)$ (MBS or DBS) is

$$R_u = B_u \log_2(1 + \gamma_{u,s(u)}), \tag{5}$$

The instantaneous rate in (5) increases with both the allocated bandwidth B_u and the SINR in (4). Where B_u is the access bandwidth allocated to user u (by equal splitting or proportional-sharing policies).

For each DBS d, the aggregate backhaul traffic carried over the MBS–DBS link is

$$R_d^{(b)} = \sum_{u \in \mathcal{U}} \alpha_{ud} R_u. \tag{6}$$

We impose a backhaul capacity constraint

$$R_d^{(b)} \le C_d^{(b)}, \quad \forall d \in \mathcal{D}. \tag{7}$$

C. Queueing and Delay Model

Each associated user u is modeled as an M/M/1 queue at its serving BS. Packets for user u arrive according to a Poisson process with rate λ_u (packets/s) and are served with rate μ_u determined by the achievable rate R_u and packet size. Denoting the utilization factor by $\rho_u = \lambda_u/\mu_u < 1$, the sojourn time (queueing plus service) in an M/M/1 queue is

$$T_u^{\text{acc}} = \frac{1}{\mu_u - \lambda_u} \tag{8}$$

Using standard M/M/1 queueing results [14], the access sojourn time in (8) provides a smooth latency signal for RL training. To account for backhaul congestion, we consider a deterministic penalty function

$$T_d^{\text{bh}} = \kappa \frac{L^{\text{bh}}}{C_d^{(b)} - R_d^{(b)}}, \quad R_d^{(b)} < C_d^{(b)}, \tag{9}$$

where L^{bh} is an average payload size and $\kappa > 0$ captures protocol overhead. This term grows rapidly as the backhaul link saturates.

The end-to-end delay for user u is then approximated as

$$D_u = T_u^{\text{acc}} + \sum_{d \in \mathcal{D}} \alpha_{ud} T_d^{\text{bh}}, \tag{10}$$

where the second term applies if user u is served via a DBS.

D. Objective, Fairness, and Penalties

Let

$$\bar{D} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} D_u \tag{11}$$

denote the mean delay, and let L_d denote the aggregate traffic load at BS d. Jain's fairness index is

$$J = \frac{\left(\sum_{b \in \mathcal{S}} \bar{\lambda}_b\right)^2}{|\mathcal{S}| \sum_{b \in \mathcal{S}} \bar{\lambda}_b^2} \tag{12}$$

We adopt Jain's fairness index in (12) to measure how evenly the traffic load is distributed across the MBS and DBSs, where $\bar{\lambda}_b$ denotes the average traffic load at BS b, and $\mathcal{S}$ is the set of all BSs (MBS + DBSs).

To provide the RL agents with explicit signals for backhaul congestion and queue instability, we define two penalty terms:

$$P_{\text{bh}} = \sum_{d \in \mathcal{D}} \max\left(0, \frac{R_d^{(b)} - C_d^{(b)}}{C_d^{(b)}}\right), \tag{13}$$

$$P_{\text{stab}} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathbf{1}_{\{\lambda_u \ge \mu_u\}}. \tag{14}$$

The first measures normalized backhaul overload, while the second is the fraction of unstable user queues.

The joint optimization problem over association decisions $\{\alpha_{ud}\}$ and DBS positions $\{\mathbf{p}_d\}$ is

$$\min_{\{\alpha_{ud}\},\{\mathbf{p}_d\}} \quad \bar{D} \tag{15}$$

$$\text{s.t.} \quad R_d^{(b)} \le C_d^{(b)}, \ \forall d, \tag{16}$$

$$\lambda_u < \mu_u, \ \forall u, \tag{17}$$

$$\sum_d \alpha_{ud} \le 1, \ \alpha_{ud} \in \{0,1\}, \ \forall u, d. \tag{18}$$

This is a mixed-integer nonlinear program (MINLP) that is difficult to solve optimally in real time, motivating the proposed RL approach.

III. Hierarchical Actor–Critic (HAC) Framework

We develop a hierarchical actor–critic framework to solve the above problem adaptively. The key idea is to separate control across two timescales:

- A *high-level* agent updates DBS positions every K association intervals based on slow-varying traffic patterns and spatial load imbalance.
- A *low-level* agent performs user association decisions at every time step based on instantaneous channel and queue states.

Both agents share a global reward that embeds delay, fairness, backhaul utilization, and queue stability.

Our design follows the standard actor–critic paradigm [15]–[18], but tailors the architecture and reward to the hierarchical nature of placement and association in drone-assisted IoT networks, in line with recent DRL/MARL trends in wireless systems [6]–[8].

A. State, Action, and Reward Design

Let t index association time steps, and k index placement decision epochs (every K steps).

1) High-Level State and Action: The high-level state at epoch k collects

$$s_k^{\text{H}} = \left(\{\mathbf{p}_d\}_{d \in \mathcal{D}}, \{L_d\}_{d \in \mathcal{S}}, \{\bar{Q}_d\}_{d \in \mathcal{S}}\right), \tag{19}$$

where $\bar{Q}_d$ denotes an aggregated queue-length indicator at BS d. The high-level action is a vector of 2-D or 3-D displacement commands:

$$a_k^{\text{H}} = \{\Delta \mathbf{p}_d\}_{d \in \mathcal{D}}. \tag{20}$$

2) Low-Level State and Action: The low-level state at time t includes

$$s_t^{\mathrm{L}} = \big(\{\gamma_{us}\}, \{\lambda_u\}, \{Q_u\}, \{\mathbf{p}_d\}, \text{backhaul headroom}\big), \quad (21)$$

where Q_u is the queue length for user u. The low-level action is a set of association decisions $\{\alpha_{ud}\}$, typically implemented as a sequential assignment or as probabilities over candidate BSs.

3) Shared Reward: Using the queueing-based delay signal from (8) and the load-balancing metric in (12), we define a shared global reward r_t for both agents:

$$r_t = -\big(\omega_1 \bar{D}_t + \omega_2(1 - J_t) + \rho_1 P_{\mathrm{stab},t} + \rho_2 P_{\mathrm{bh},t}\big), \quad (22)$$

where $\omega_1, \omega_2, \rho_1, \rho_2$ are non-negative weights; $\bar{D}_t$ is the empirical mean delay over users in slot t; J_t is Jain's fairness index; and $P_{\mathrm{stab},t}$ and $P_{\mathrm{bh},t}$ reuse the stability and backhaul penalties introduced in Section II. In our experiments we set $(\omega_1, \omega_2, \rho_1, \rho_2) = (1.0, 0.5, 2.0, 1.0)$.

B. Training Algorithm

We adopt an off-policy actor–critic scheme with replay buffers and target networks for both levels. The high-level actor/critic are updated every K steps using aggregated transitions, while the low-level networks are updated at every step, following best practices from deep RL [15]–[18].

C. Complexity and Scalability Analysis

Let $|\mathcal{U}|$ and $|\mathcal{D}|$ denote the numbers of users and DBSs, respectively. At each low-level step, computing SINR and rates for every user–BS pair scales as $\mathcal{O}(|\mathcal{U}||\mathcal{D}|)$. A forward/backward pass of the low-level networks has complexity $\mathcal{O}(N_{\mathrm{L}})$, where N_{L} is the number of parameters. The high-level networks are updated only every K steps, so their amortized cost is $\mathcal{O}(N_{\mathrm{H}}/K)$.

Overall, the per-step complexity of HAC is linear $|\mathcal{U}||\mathcal{D}|$ for fixed network sizes, and the hierarchical decomposition avoids the combinatorial explosion that would arise from a monolithic state–action space. In Section IV, we empirically study how latency and convergence behave as $|\mathcal{U}|$ and $|\mathcal{D}|$ increase.

Algorithm 1 Training procedure for the hierarchical actor–critic (HAC) framework

Initialize high-level actor/critic $(\pi^{\mathrm{H}}, Q^{\mathrm{H}})$ and low-level $(\pi^{\mathrm{L}}, Q^{\mathrm{L}})$
Initialize replay buffers $\mathcal{B}^{\mathrm{H}}, \mathcal{B}^{\mathrm{L}}$
for each episode **do**
  Reset environment, initialize DBS positions
  for $t = 0, 1, \ldots, T-1$ **do**
    if $t \bmod K == 0$ **then**
      Observe s_k^{H}; sample $a_k^{\mathrm{H}} \sim \pi^{\mathrm{H}}(\cdot|s_k^{\mathrm{H}})$
      Move DBSs according to a_k^{H}
    end if
    Observe low-level state s_t^{L}
    Sample association action $a_t^{\mathrm{L}} \sim \pi^{\mathrm{L}}(\cdot|s_t^{\mathrm{L}})$
    Apply a_t^{L}, evolve queues and rates, compute reward r_t in (22)
    Store transition $(s_t^{\mathrm{L}}, a_t^{\mathrm{L}}, r_t, s_{t+1}^{\mathrm{L}})$ in $\mathcal{B}^{\mathrm{L}}$
    if $t \bmod K == K-1$ **then**
      Aggregate rewards and terminal high-level state to form $(s_k^{\mathrm{H}}, a_k^{\mathrm{H}}, \tilde{r}_k, s_{k+1}^{\mathrm{H}})$
      Store $(s_k^{\mathrm{H}}, a_k^{\mathrm{H}}, \tilde{r}_k, s_{k+1}^{\mathrm{H}})$ in $\mathcal{B}^{\mathrm{H}}$
    end if
    Update low-level networks using minibatches from $\mathcal{B}^{\mathrm{L}}$
    if $t \bmod K == K-1$ **then**
      Update high-level networks using minibatches from $\mathcal{B}^{\mathrm{H}}$
    end if
  end for
end for

IV. Performance Evaluation

A. Simulation Setup

We consider a $1000\,\mathrm{m} \times 1000\,\mathrm{m}$ macrocell with a single MBS located at the center at altitude $30\,\mathrm{m}$ and $|\mathcal{D}| = 3$ DBSs deployed within the same horizontal area. Unless otherwise stated, we set $|\mathcal{U}| = 40$ IoT users, whose locations are drawn independently and uniformly over the area and remain fixed within each episode.

User packet arrivals follow independent Poisson processes with rates λ_u drawn uniformly from $[20, 60]$ packets/s. Each packet has size $L = 8 \times 10^3$ bits (1 kB). The MBS and DBS transmit powers are 40 dBm and 30 dBm, respectively, over a system bandwidth of $B = 10$ MHz and noise spectral density $N_0 = -174$ dBm/Hz. DBS altitudes are initialized uniformly in $[120, 250]$ m and kept within this range during training.

We independently draw each DBS backhaul capacity $C_d^{(b)}$ and uniformly from $[120, 220]$ Mbit/s at the beginning of each episode. These capacity values jointly determine the M/M/1 access queues, backhaul loads, and penalties P_{bh} and P_{stab} used in the reward.

Each episode consists of $T = 40$ association intervals. The low-level association policy acts at every interval, while the high-level placement policy updates every $K = 10$ intervals.

B. RL Training and Evaluation Protocol

Both HAC and the flat RL baseline use deterministic actor–critic policies implemented by two-layer multilayer perceptrons (MLPs) with 128 hidden units per layer and ReLU activations. The actor output is squashed by a $\tanh$ function and perturbed with zero-mean Gaussian exploration noise ($\sigma = 0.2$), with actions clipped to $[-1, 1]$. We employ a DDPG-style update [17] with learning rates 10^{-3} for both actor and critic networks, discount factor $\gamma = 0.95$, and Polyak averaging coefficient $\tau = 0.01$ for the target networks.

For each method we run $N_{\mathrm{seed}} = 10$ independent random seeds, where each seed corresponds to a different initialization of user locations, arrival rates, backhaul capacities, and network weights. Each run is trained for $N_{\mathrm{ep}} = 400$ episodes. For seed m and episode index k, let $z_{m,k}$ denote the per-episode

average of a performance metric z (reward, delay, or fairness). We report the sample mean

$$\bar{z}_k = \frac{1}{N_{\text{seed}}} \sum_{m=1}^{N_{\text{seed}}} z_{m,k}$$

and an approximate 95% confidence interval (CI)

$$\text{CI}_k = 1.96 \, \frac{\text{std}\{z_{m,k}\}}{\sqrt{N_{\text{seed}}}}.$$

Learning curves show $\bar{z}_k \pm \text{CI}_k$ across episodes. For steady-state summaries we average each metric over the last 50 episodes of each run and then average across seeds.

C. Baselines and Metrics

We focus on three methods:

- **HAC**: the proposed hierarchical actor–critic framework.
- **Flat RL**: a single-level DDPG agent that jointly controls DBS displacements and association biases with a combined state and action space.
- **Greedy**: a non-learning baseline with fixed DBS positions and SINR-based user association (users attach to the BS with the strongest instantaneous SINR), ignoring backhaul and queueing penalties.

We evaluate:

1) Average end-to-end delay $\bar{D}$ defined in Section II.
2) Jain's fairness index J of the BS loads.
3) Convergence speed and stability of the learning curves.

All reported results are averaged over the independent seeds with 95% CIs as described above.

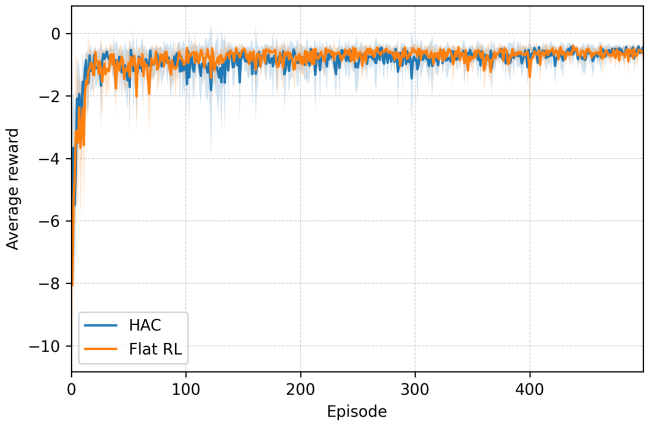
D. Main Results: Learning Curves and Steady-State Performance

Fig. 1(a) shows the learning curves of HAC and flat RL in terms of the global reward. Both methods quickly improve the average reward from large negative values to near zero within the first tens of episodes. HAC converges slightly faster and exhibits lower variance in the early training phase, reflecting the reduced effective action space from the hierarchical decomposition.

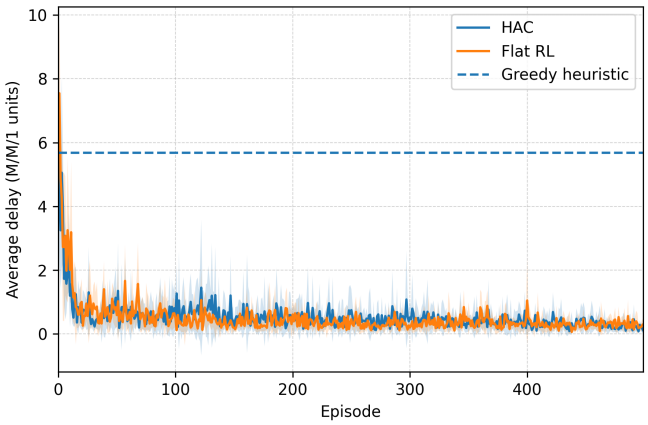
Fig. 1(b) reports the corresponding average delay during training. The greedy baseline is shown as a horizontal dashed line. Both RL methods dramatically reduce the delay relative to the greedy SINR policy. HAC attains slightly lower steady-state delay than flat RL, while maintaining stable convergence across seeds.

To highlight steady-state behavior, we summarize the last 50 episodes of each run and then average across seeds. The resulting values in Table II compare the greedy, flat RL, and HAC methods in terms of average delay and Jain fairness.

HAC and flat RL both reduce the average delay by more than an order of magnitude relative to the greedy policy, with HAC yielding the lowest delay among the three. Both RL methods also improve Jain's index compared to greedy SINR association, and HAC provides a small yet consistent advantage in delay without sacrificing fairness. These results



(a) Average reward



(b) Average delay

Fig. 1. Training performance of HAC and flat RL in the drone-assisted IoT network (mean ± 95% CI over 10 random seeds). Dashed lines indicate the greedy SINR-based baseline where applicable.

TABLE II
STEADY-STATE PERFORMANCE (LAST 50 EPISODES; MEAN ± STANDARD DEVIATION ACROSS SEEDS)

Method	Avg. delay (M/M/1 units)	Jain fairness J
Greedy	5.682 ± 4.290	0.253 ± 0.007
Flat RL	0.303 ± 0.358	0.340 ± 0.034
HAC	0.260 ± 0.364	0.339 ± 0.035

confirm that the hierarchical decomposition provides robust benefits while maintaining stable and reproducible learning behavior across random seeds.

Overall, the performance evaluation indicates that HAC yields consistent latency and fairness gains across a range of traffic and backhaul conditions while remaining robust to architectural and hyperparameter choices.

V. Discussion and Future Work

A. Modeling Choices and Limitations

We use M/M/1 queues to model access-side delay mainly for analytical simplicity and compatibility with RL training. Although IoT traffic can be bursty and correlated, such

approximations are widely used in resource allocation and scheduling [14], and in our setting the sojourn time T_u^{acc} in (8) provides a smooth latency signal that works well with actor–critic updates. A natural extension is to incorporate richer traffic models (e.g., Markov-modulated Poisson or trace-driven arrivals) for ultra-reliable or heavily bursty scenarios.

B. Scalability and Towards Multi-Agent RL

The current HAC design assumes a logically centralized controller with access to global state, which is reasonable for mid-scale edge deployments but less so for very large networks. Our experiments varying $|\mathcal{U}|$ and $|\mathcal{D}|$ indicate that HAC preserves a consistent performance gap over flat RL and greedy association as the system scales, suggesting that the hierarchical decomposition is a good basis for more distributed designs. Future work will explore decentralized or multi-agent formulations (e.g., CTDE) in which DBSs act as local agents exchanging compact summaries, potentially supported by graph-based aggregation [6], [9], [10].

C. Implementation Aspects

From a deployment standpoint, HAC can be trained offline on realistic traces and fine-tuned online. Lightweight actor networks can run on an edge controller co-located with the MBS, while DBSs execute simple motion and association commands. Our sensitivity studies (not shown in detail) suggest that the method is robust to reasonable variations of learning rates, discount factor γ, and soft-update parameter τ, which is encouraging for integration into existing software-defined RAN or edge orchestration platforms.

VI. Conclusion

We proposed a hierarchical actor–critic framework for drone-assisted IoT networks that jointly optimizes DBS placement and user association under backhaul and queueing constraints. By separating slow-timescale placement from fast-timescale association and coordinating both via a shared reward encoding delay, fairness, and stability, HAC provides a flexible way to exploit UAV mobility while protecting backhaul resources.

We presented the system model, M/M/1-based delay formulation, RL architecture, and complexity analysis, and we evaluated HAC against greedy SINR-based association and a flat RL baseline over multiple random seeds. Simulations show that HAC reduces average delay by more than an order of magnitude and improves Jain fairness compared to greedy association, while slightly outperforming flat RL and exhibiting more stable learning. Additional experiments indicate that these gains persist as the number of users and DBSs increases and that variants without a shared global reward tend to sacrifice fairness.

Future work will focus on decentralized multi-agent extensions with CTDE, more realistic and bursty traffic models, and integration with practical edge and software-defined RAN platforms, moving HAC closer to real-world drone-assisted IoT deployments.

Acknowledgment

This work is supported by the Research Center for Distributed Resilient and Emergent-Intelligence-Based Additive Manufacturing (DREAM) under NSF E-RISE RII Award OIA-2417062.

References

[1] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[2] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 36–42, 2016.

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[4] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 2706–2716, 2013.

[5] N. Sapountzis, T. Spyropoulos, N. Nikaein, and U. Salim, "User association in hetnets: Impact of traffic differentiation and backhaul limitations," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3358–3371, 2017.

[6] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1226–1252, 2021.

[7] Y. Qin, Z. Zhang, X. Li, W. Huangfu, and H. Zhang, "Deep reinforcement learning based resource allocation and trajectory planning in integrated sensing and communications UAV network," *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 8158–8169, 2023.

[8] C. Dai, K. Zhu, and E. Hossain, "Multi-agent deep reinforcement learning for joint decoupled user association and trajectory design in full-duplex multi-UAV networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 6056–6070, 2023.

[9] I. Alablani, A. Alshamrani, S. Alshamrani, and F. Alshamrani, "DQN-GNN-based user association approach for wireless networks," *Mathematics*, vol. 11, no. 20, p. 4286, 2023.

[10] Z. Cheng, M. Liwang, N. Chen, L. Huang, N. Guizani, and X. Du, "Learning-based user association and dynamic resource allocation in multi-connectivity enabled unmanned aerial vehicle networks," *Digital Communications and Networks*, vol. 10, no. 1, pp. 53–62, 2024.

[11] H. Chen, J. Miao, R. Wang, H. Li, and X. Zhang, "A hierarchical deep reinforcement learning approach for throughput maximization in reconfigurable intelligent surface-aided unmanned aerial vehicle–integrated sensing and communication network," *Drones*, vol. 8, no. 12, p. 717, 2024.

[12] F. n. Kim *et al.*, "HiMAQ: Hierarchical multi-agent Q-learning-based throughput and fairness improvement for UAV-aided IoT networks," *Journal of Network and Computer Applications*, 2024, early access.

[13] A.-H. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569–572, 2014.

[14] L. Kleinrock, *Queueing Systems, Volume I: Theory*. New York, NY, USA: John Wiley & Sons, 1975.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[16] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel *et al.*, "Continuous control with deep reinforcement learning," in *Proc. International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.