CCL Winter Camp 2024

Reinforcement Learning for Aerial Base Station Deployment in Satellite-Terrestrial Networks

Linh T. Hoang

Urabandai, Feb 26, 2023

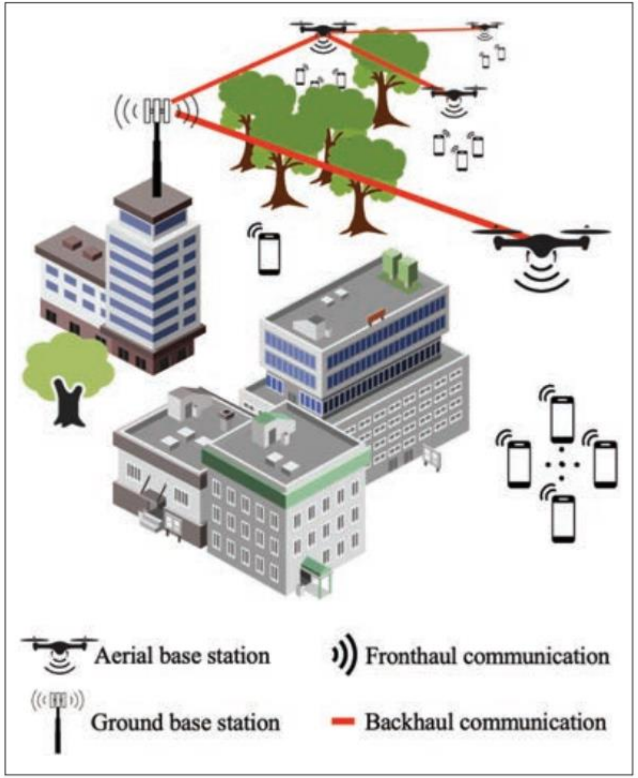
Contents

- UAV-mounted Base Stations (UAV-BSs)
- Single-UAV deployment via A2C Algorithm
- Multi-UAV deployment via Multi-Agent RL
- Conclusions

UAV-mounted Base Stations

- A promising technology towards high-quality services and ubiquitous coverage
- **Potential use cases**:
    - Remote areas
    - Large open-air events (concerts, carnivals, etc)
    - Natural disaster-affected areas
- **Challenge:**

  The UAV-BSs **automatically adjust their locations** over time in response to dynamics of the network situation (e.g., the user's spatial distribution)



ABSs connect with Ground Terminals (fronthaul connection) and the terrestrial infrastructure (backhaul connection) [1]

[1] P. Q. Viet and D. Romero, "Aerial Base Station Placement: A Tutorial Introduction," IEEE Commun. Mag., vol. 60, no. 5, pp. 44–49, May 2022.

Single-UAV Deployment

UAV-aided Satellite-Terrestrial Networks

Network scenario:

- One UAV-BS is deployed to complement the terrestrial base station (macro BS).
- The UAV-BS hold backhaul links with both the LEO satellite and the macro BS.
- Ground users are either served by the UAV-BS or the macro BS.

Problem Statement:

- Given a random distribution of users with unsatisfied data rates
- How the UAV-BS can complement the macro BS to provide better rates for this user set.

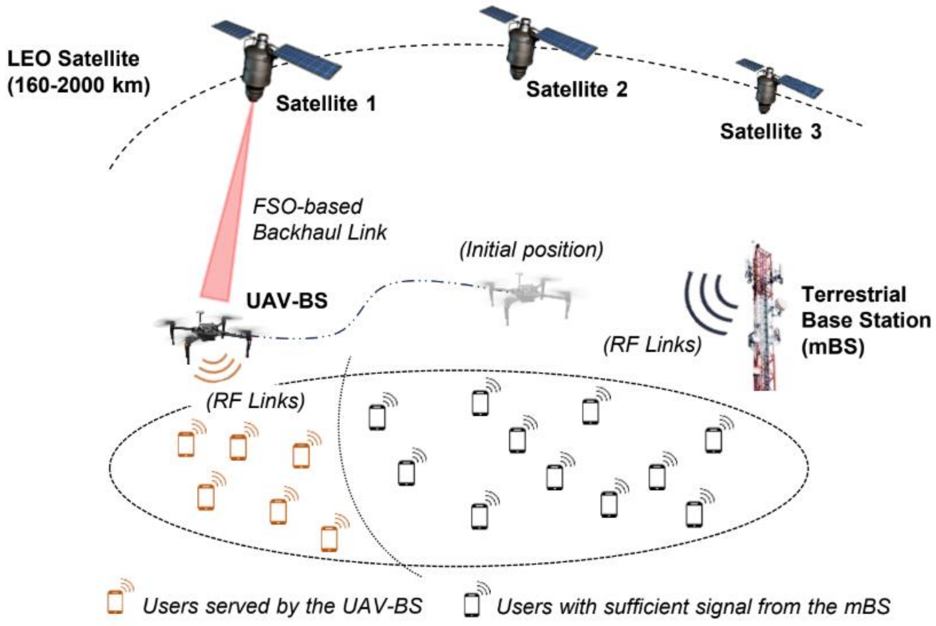


Figure 1: UAV-assisted Satellite-terrestrial Network

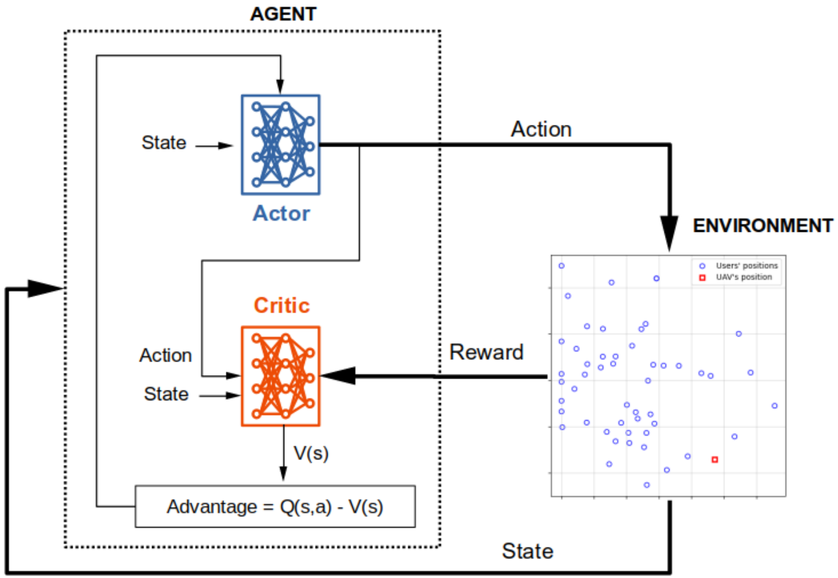
Training with A2C Algorithm (1/2)

- **State**: the users' and UAV-BS's coordinates

- **Actions**: moving with 5 possible directions
  (1) northward,
  (2) westward,
  (3) southward,
  (4) eastward,
  (5) remain stationary (no movement)

- **Reward**:
  +1 if d(t+1) > d(t),
  -1 if d(t+1) < d(t),
  -0.1 if d(t+1) = d(t)

(d(t): the average data rate of all users at time step t)



Training with A2C Algorithm (2/2)

A2C = Advantage Actor-Critic [1],
a policy-based RL algorithm

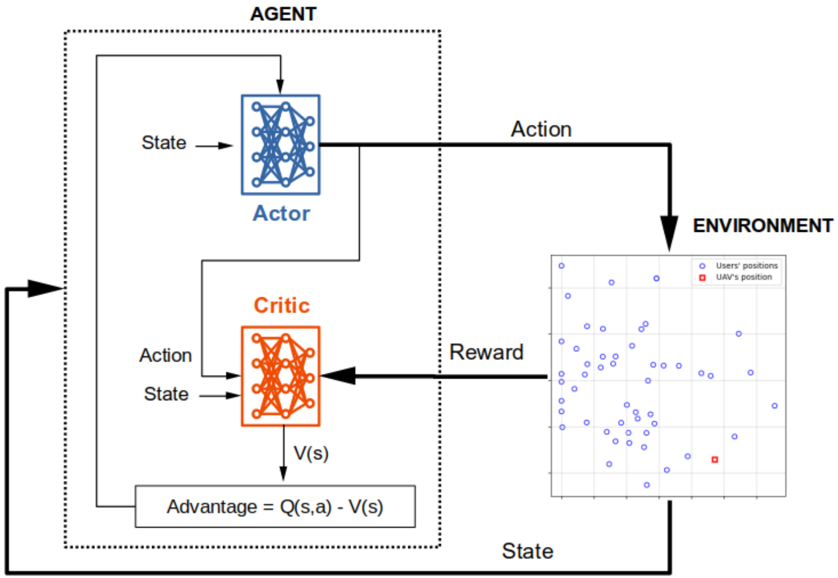
Actor: outputs logits for a categorical probability distribution over all possible actions.

- In training: update the actor's params to maximize the advantage function.

Critic: estimates the state-value function of the environment's state.

- In training: update the critic's params to minimize the difference between the observed and the predicted value functions.

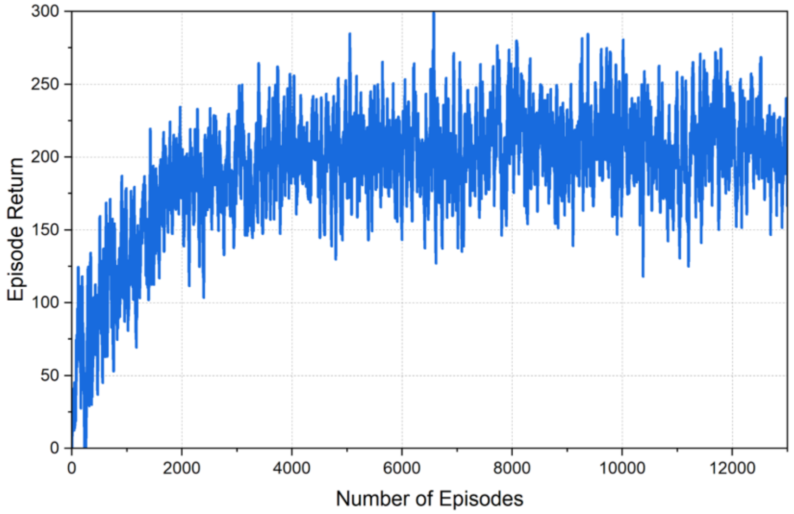
Advantage function: how much better it is to take a specific action compared to the average, general action at the given state



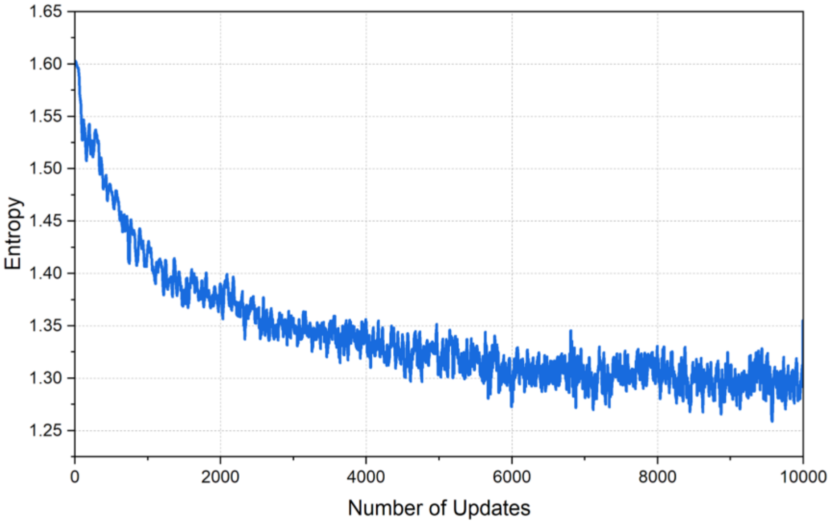
[1] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in Proceedings of the 33rd International Conference on Machine Learning (ICML), vol. 48, 2016, p. 1928–1937.

A2C Algorithm: Training Results

The agent gradually forms better movement policy for the UAV-BS with higher rewards



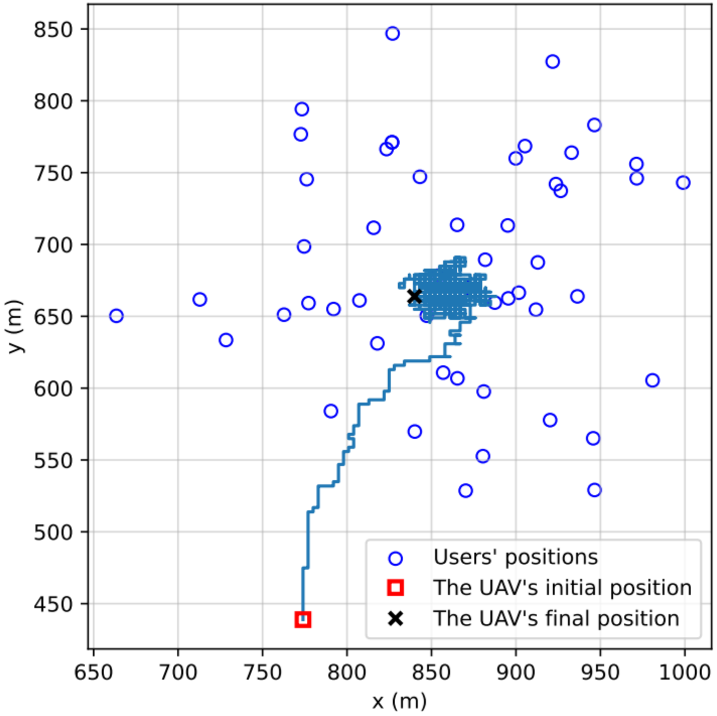
The action selected by the agent gradually becomes less random (i.e., more intentional)



A2C Algorithm: Testing Results

To demonstrate the generalization ability of the agent after training, a trained A2C agent is tested to control the UAV-BS's movements in one single episode.

- In the test environment, the spatial distribution of users and the initial location of the UAV were set up randomly and not previously seen by the agent.
- The agent was not trained during the test.

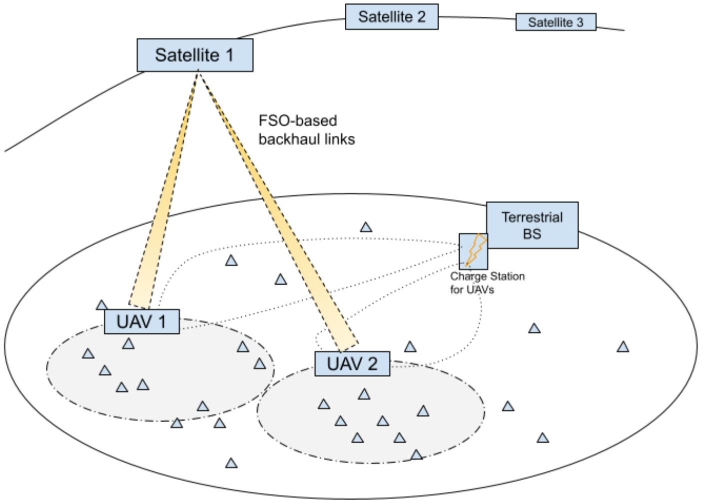


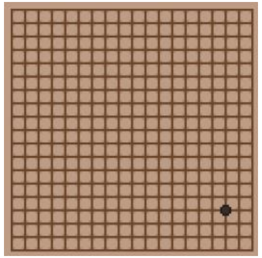
Behavior of an A2C agent after 4 hours of training

Multi-UAV Deployment

Multi-UAV Deployment

- **Satellite-Air-Ground Integrated Networks (SAGIN)**: multiple UAV-BSs are deployed to complement the terrestrial BS.
- Take into account constraints of the **FSO-based backhaul links** with LEO satellites.
- Multiple UAV-BSs are expected to cooperate to efficiently serve the ground users → **Multi-agent RL**





Multi-agent RL for playing Go

Multi-Agent Reinforcement Learning

Current setups:

- One single macro BS in the top-left corner.
- Three UAV-BSs cooperate to support the macro BS in providing high data rates for the ground users.
- The RL agent learns directly from image pixels and are required to cooperate with each other and with the macro BS.
- Simulation results: not yet available.

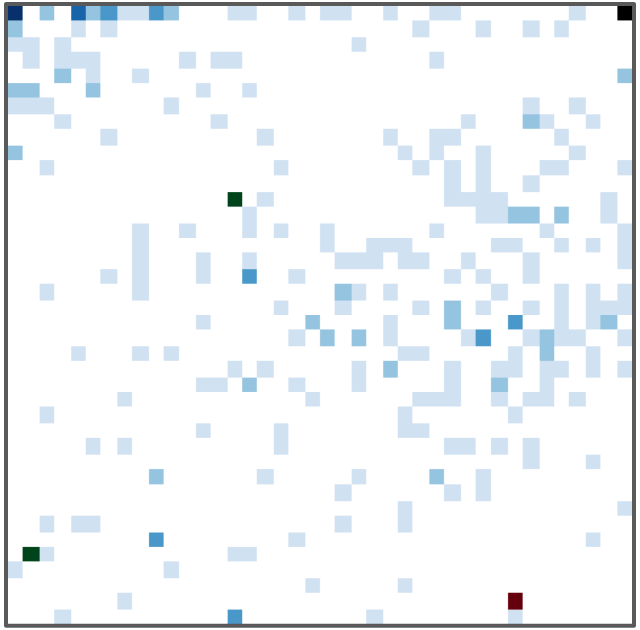


Illustration of the learning environment

References

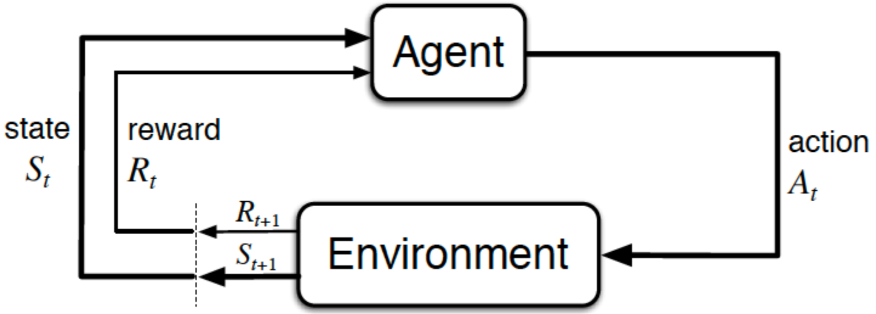
1. R. S. Sutton and A. G. Barto, Reinforcement Learning, 2nd ed. The MIT Press, 2018.
2. V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
3. J. Liu, Y. Shi, Z. Md. Fadlullah, and N. Kato, “Space-Air-Ground Integrated Network: A Survey,” *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, pp. 2714–2741, 2018.
4. A. Feriani and E. Hossain, “Single and Multi-Agent Deep Reinforcement Learning for AI-Enabled Wireless Networks: A Tutorial,” *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 1226–1252, 2021.

Thank you for your attention!

Appendix

RL Formulation

- An **agent** learning to interact with its **environment**.
- At each time step, the agent receives the environment's **state**, and the agent must choose an appropriate **action** in response.
- One time step later, the agent receives a **reward** (the environment indicates whether the agent has responded appropriately to the state) and a new state.
- The agent aim to maximize the **expected cumulative reward** (i.e., the expected sum of rewards attained over all time steps).



The agent-environment interaction in reinforcement learning.
(Sutton and Barto, 2017)

State-Value Function and Bellman Equation

State-Value Functions

- The **state-value function** for a policy π is denoted v_π. For each state $s \in \mathcal{S}$, it yields the expected return if the agent starts in state s and then uses the policy to choose its actions for all time steps. That is, $\boxed{v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s]}$. We refer to $v_\pi(s)$ as the **value of state s under policy π**.

The discounted return (cumulative reward) at time *t*:

$$G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots.$$

Bellman Equations

- The **Bellman expectation equation for** v_π is: $\boxed{v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]}$.

Optimality

- A policy π' is defined to be better than or equal to a policy π if and only if $v_{\pi'}(s) \geq v_\pi(s)$ for all $s \in \mathcal{S}$.

Action-Value Function and Optimal Policies

Action-Value Functions

- The **action-value function** for a policy π is denoted q_π. For each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, it yields the expected return if the agent starts in state s, takes action a, and then follows the policy for all future time steps. That is, $\boxed{q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a]}$. We refer to $q_\pi(s,a)$ as the **value of taking action a in state s under a policy π** (or alternatively as the **value of the state-action pair s, a**).
- All optimal policies have the same action-value function q_*, called the **optimal action-value function**.

Optimal Policies

- Once the agent determines the optimal action-value function q_*, it can quickly obtain an optimal policy π_* by setting $\boxed{\pi_*(s) = \arg\max_{a \in \mathcal{A}(s)} q_*(s,a)}$.

The discounted return (cumulative reward) at time *t*:

$$G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots.$$

The problem now is how to estimate the optimal value function q*(s,a)