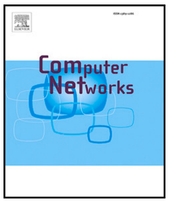
Computer Networks 254 (2024) 110789

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Joint path planning and power allocation of a cellular-connected UAV using apprenticeship learning via deep inverse reinforcement learning

Alireza Shamsoshoara [a,1], Fatemeh Lotfi [b,*,1], Sajad Mousavi [c], Fatemeh Afghah [b], İsmail Güvenç [d]

[a] School of Informatics, Computing and Cyber Systems at Northern Arizona University, Flagstaff, AZ, USA
[b] Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA
[c] Harvard Medical School, Boston, MA, USA
[d] North Carolina State University, Raleigh, NC, USA

ARTICLE INFO

Keywords:
Apprenticeship learning
Cellular-connected drones
Inverse reinforcement learning
Path planning
UAV communication

ABSTRACT

This paper investigates an interference-aware joint path planning and power allocation mechanism for a cellular-connected unmanned aerial vehicle (UAV) in a sparse suburban environment. The UAV's goal is to fly from an initial point and reach a destination point by moving along the cells to guarantee the required quality of service (QoS). In particular, the UAV aims to maximize its uplink throughput and minimize interference to the ground user equipment (UEs) connected to neighboring cellular base stations (BSs), considering both the shortest path and limitations on flight resources. Expert knowledge is used to experience the scenario and define the desired behavior for the sake of the agent (i.e., UAV) training. To solve the problem, an apprenticeship learning method is utilized via inverse reinforcement learning (IRL) based on both Q-learning and deep reinforcement learning (DRL). The performance of this method is compared to learning from a demonstration technique called behavioral cloning (BC) using a supervised learning approach. Simulation and numerical results show that the proposed approach can achieve expert-level performance. We also demonstrate that, unlike the BC technique, the performance of our proposed approach does not degrade in unseen situations.

1. Introduction

The rapid advancement of unmanned aerial vehicles (UAVs) has spurred extensive research in both academia and industry. UAVs are being deployed in a variety of applications, including cargo delivery, search and rescue (SAR), aerial imaging, disaster monitoring, surveillance, and public safety [1–8]. These versatile applications leverage unique UAV capabilities such as a wide ground field of view (FoV), 3-dimensional movement, fast deployment, and agile response. One significant area of interest is drone-assisted communication, where UAVs act as aerial base stations (BSs) or flying access points (APs) to support terrestrial user equipment (UE) connectivity, particularly in disaster relief scenarios. UAVs can extend cellular coverage and improve connectivity, and they can relay information between UEs and neighboring BSs, enhancing the overall communication infrastructure [9–14].

As UAV technology evolves, their integration into cellular networks, including 5G and beyond, presents several challenges. These challenges include interference management, spectrum management, quality-of-service (QoS) management, frequent handovers, and ensuring robust, secure, and reliable communication [15–23]. Traditional cellular infrastructures, designed primarily for terrestrial UEs, need significant modifications to serve aerial UEs effectively [24–27]. Advanced techniques such as massive multiple-input multiple-output (mMIMO) technology and beamforming algorithms have been proposed to enhance interference management and improve signal targeting. These methods aim to direct the signal towards the intended receiver and minimize interference with other users, thereby ensuring reliable communication for aerial UEs [28–30].

Most existing research employs model-free reinforcement learning (RL) methods, such as temporal difference learning, to address complex issues like interference and path planning [31–46]. These methods, while powerful, often struggle with dynamic and complex environments where problems are non-convex and computationally intensive. The effectiveness of RL methods depends heavily on the definition of the reward function, which establishes the relationship between the Markov Decision Process (MDP) environment, Q-action values, and optimal policy learning. As the complexity of the problem

* Corresponding author.
E-mail addresses: alireza_shamsoshoara@nau.edu (A. Shamsoshoara), flotfi@clemson.edu (F. Lotfi), seyedsajad_mousavi@hms.harvard.edu (S. Mousavi), fafghah@clemson.edu (F. Afghah), iguvenc@ncsu.edu (İ. Güvenç).
[1] These authors contributed equally to this work

https://doi.org/10.1016/j.comnet.2024.110789
Received 21 February 2024; Received in revised form 3 August 2024; Accepted 4 September 2024
Available online 12 September 2024

increases, defining a comprehensive reward function becomes challenging, leading to suboptimal policies. Furthermore, in real-world scenarios, defining a reward function that captures all aspects of the problem is not always feasible, resulting in less meaningful comparisons with other methods.

1.1. Motivation

Traditional RL approaches face significant hurdles in dynamic and complex environments due to the difficulty of formulating comprehensive reward functions. This often results in suboptimal policy outcomes, especially when dealing with non-convex problems. Moreover, most existing methods do not effectively leverage expert knowledge, which can be crucial for developing more robust and adaptive UAV path planning and power allocation strategies. With the rise of open-source software, simulators, and emulators such as Network Simulator-3 (NS3) [47], srsLTE [48], OpenAirInterface [49], and others, it is now possible to generate optimal behavior for an agent in a simulated environment with expert assistance. This data can be used for the agent to learn optimal behavior through imitation learning (IL), learning from demonstration (LfD), or apprenticeship learning (AL). In [50], the authors examined a disaster relief scenario where a UAV assists terrestrial UEs in scheduling packet deliveries to neighboring BSs using an IL solution. The authors implemented a supervised learning solution in the form of a behavioral cloning (BC) approach to mimic the expert's behavior in a similar situation. BC typically does not consider any reward functions and relies solely on visited states and actions, with the agent mindlessly following the expert. If the expert makes incorrect decisions, the agent may replicate those errors. Moreover, suppose the state visited by the agent has never been experienced by the expert. In that case, the agent cannot take appropriate action since that state was not considered in the optimal learned policy.

1.2. Novelty and contribution

To overcome these limitations, we propose a novel approach based on apprenticeship learning via inverse reinforcement learning (IRL) [51,52]. This method leverages expert-generated data to reconstruct optimal reward functions, enabling the UAV to learn effective policies without needing a meticulously defined reward function. By employing deep Q-networks (DQN), our approach demonstrates superior performance in various scenarios, including those not encountered during training. Our significant contributions are:

- Novel Problem Formulation: We present a comprehensive formulation of the joint path planning and power allocation problem for a cellular-connected UAV. The objective is to minimize uplink (UL) transmission interference to terrestrial UEs while maximizing the UAV's UL throughput and signal-to-noise ratio (SNR). This formulation considers multiple factors such as UAV transmission power, path planning, and the density of terrestrial UEs, ensuring a holistic approach to the problem.
- Open-Source Simulator: We develop an open-source simulator to generate expert trajectories and behaviors in a controlled environment. This simulator is fully object-oriented and customizable, allowing users to adjust parameters such as the number of cells and UE density. A graphical user interface (GUI) visually represents the UAV's interaction with the simulation environment. This tool not only aids in training but also serves as a valuable resource for future research.
- Apprenticeship Learning Framework: We design an apprenticeship learning framework using IRL and deep IRL. This approach analyzes expert behaviors to derive an optimal reward function, which is then used to generate effective policies for the UAV. The framework utilizes the strengths of Q-learning and DQN to handle both simple and complex scenarios effectively.
- Comprehensive Performance Evaluation: We conduct a thorough evaluation of our apprenticeship learning approach, comparing it with alternative methods such as behavioral cloning, shortest path, and random path strategies. Our results demonstrate the robustness and effectiveness of our approach in both familiar and unseen scenarios. We show that our method can achieve expert-level performance and maintain its robustness in dynamic and unpredictable environments.

Furthermore, our method enhances the decision-making capabilities of UAVs in urban air mobility applications, where safety and adaptability are critical. By balancing multiple factors, such as throughput, coverage, energy consumption, and interference, our approach ensures reliable and efficient UAV operations. This has significant implications for real-world applications, particularly in densely populated urban areas where UAVs must navigate complex environments.

1.3. Organization

The rest of this paper is organized as follows. Section 2 reviews the related works regarding the cellular-connected UAV for similar problems and approaches. The problem definition and the system model are discussed in Section 3 with all related assumptions. Section 4 provides background information on inverse RL approach. Section 5 introduces the apprenticeship learning via IRL using both Q-learning with linear function approximation and Deep Q-Network. Section 6 evaluates the convergence and other metrics and parameters using our designed simulator. Conclusions and discussions are summarized in Section 7.

1.4. Table of notations

Table 2 shows all the parameters and notations used in this study in both the system model and methodology.

2. Related works

Investigating cellular-connected UAVs, we delve into prior research tackling various challenges such as interference management, power allocation, task allocation, path planning, etc. These problems were tackled using both centralized and distributed approaches [30,35–44, 50,53–60]. In the context of UAV operations, autonomous navigation in unfamiliar environments stands out as a crucial task. A vital aspect of this is path planning, which essentially means figuring out the best route from the starting point to the desired destination while avoiding collisions. These path planning methods can be broadly categorized into two groups: those that rely on conventional optimization techniques and those that leverage artificial intelligence (AI) and machine learning (ML) to identify optimal routes.

2.1. Conventional optimization solutions

Conventional methods have primarily focused on optimization techniques to address UAV-related challenges:

2.1.1. Interference and power coordination

In [56], the focus was on addressing the complexities of inter-cell interference and power coordination in the context of cellular-connected UAVs. The authors tried to solve the problem of inter-interference minimization using a centralized approach. The authors assumed a central point exists to collect global information for the optimization problem. The challenge is defined as a non-convex optimization problem. Hence, the authors used a successive convex approximation (SCA) to solve the problem. They obtained a sub-optimal solution for this approach. Later, the authors proposed a decentralized approach by only using local information. While the literature shows significant results, traditional path-planning algorithms have limitations in complex and

Table 1
Literature review for cellular connected UAV trajectory planning.

Ref.	Objective	Method	Interference	Throughput	Power
[37]	Minimize the weighted sum of mission completion time and communication outage duration	DDQN	✓	×	×
[38]	Minimize UAV mission time and disconnection while ensuring cellular network connectivity	RL-TD	✓	×	×
[39]	Minimize fuel consumption and distance to the target point	DRQN	×	×	×
[40]	Minimize distance to target without collisions	A2C	×	×	✓
[41,42]	Maximize energy efficiency, minimize latency, and reduce ground network interference	DRL-ESN	✓	✓	✓
[43]	Minimize travel distance, maximize connection quality	Heuristic algorithm	×	✓	×
[44]	Maximize UAV data collection within energy and coverage limits	HAS-DQN	✓	✓	✓
[56]	Maximize weighted sum-rate for UAV and ground users, considering power and cell associations in uplink	SCA	✓	✓	✓
[57]	Minimize propulsion energy while ensuring required sensing resolutions on landmarks	SCA	×	×	✓
[58]	Minimize UAV mission time while meeting connectivity requirement	Iterative alg. and graph theory	×	✓	×
[59]	Minimize UAV Mission Completion Time while satisfying the SNR constraint	Convex opt. and graph theory	×	✓	×
[60]	Minimizes disconnection duration with reduced computational complexity	Approximation solution based on dynamic programming	×	✓	×
[61]	Maximize UAV's penetration capability in dense radar environments	PPO	✓	×	✓
[62]	Minimize energy consumption and execution latency	MARL	×	×	×
[63]	Maximize system utility	Soft Actor-Critic (SAC)	×	✓	✓
[64]	Minimize content retrieving delay and maximize content hit ratio	Multi-Agent Proximal Policy Optimization (MAPPO)	✓	✓	×
[65]	Minimizing UAV operation energy, sensing power, and reporting power	Multi-Agent Deep Deterministic Policy Gradient (MADDPG)	×	×	×
[66]	Maximize system throughput and minimize latency	PPO and DQN	✓	✓	✓
This work	Maximize UAV throughput, minimize interference, and jointly optimize power allocation and path planning	Apprenticeship learning via inverse RL - DQN	✓	✓	✓

dynamic environments. Therefore, recent studies utilize AI approaches to investigate optimal solutions in complex environments.

Moreover, [57] addresses the path planning problem by transforming the optimization problem into a convex form. The solution involves employing successive convex optimization, and block coordinate descent techniques. Furthermore, [58] achieves path planning through an iterative trajectory optimization algorithm. It incorporates graph theory to address all UAV-BS associations and chooses the most suitable candidate based on topological relationships. Also, in [59], the authors address the trajectory optimization challenge in a cellular-connected UAV scenario, ensuring continuous wireless connectivity with a ground base station (GBS). They employ convex optimization and graph theory to introduce an algorithm to approximate a UAV trajectory planning solution. In another study [60], authors introduce a dynamic programming method to optimize the UAV trajectory, considering an endurance constraint. While their approach provides an approximate solution, it involves higher computational complexity.

2.2. *AI based solutions*

AI and ML approaches have been employed to overcome the limitations of conventional methods, offering more flexibility and adaptability. Recent studies such as [37–44] model cellular-connected UAV controlling challenges as an MDP optimization problem, then utilized RL based approaches to find an optimal solution. However, these methods often struggle with scalability issues and extensive training data requirements.

2.2.1. *Reinforcement learning approaches*

Authors in [37] considered a scenario where a cellular-connected UAV aims to finish its task and be covered by the ground cellular network. They used a dueling double deep Q-network (DDQN) and RL for simultaneous navigation and radio mapping, enhancing the UAV's aerial coverage while minimizing the timeout probability (P_{out}). Similarly, [38] defined a path planning problem for a cellular-connected UAV to minimize its mission completion time and maximize the connectivity probability to the cellular BSs. They modeled the problem in a MDP environment and used a temporal difference (TD) approach to find the optimal policy for path planning optimization. [39] utilized a modified Deep Recurrent Q-Network (DRQN) algorithm combining Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to improve learning and action selection strategies for UAV path planning. They designed a reward function incorporating domain knowledge such as fuel consumption and distance to the target point. Authors in [40] employed an Advantage-Actor Critic (A2C) RL algorithm to define a path planning solution for UAVs. Furthermore, [41,42] presented a deep RL algorithm incorporating echo state network (ESN) cells. UAVs optimized energy efficiency, reduced wireless transmission latency and interference, and minimized the time needed to reach their destination using ESN cells.

2.2.2. *Deep RL and heuristic approaches*

Authors in [41,42] integrated deep RL with echo state network (ESN) cells, optimizing energy efficiency and path planning. However, these methods face convergence issues and require extensive

Table 2
Notation descriptions.

Notation	Description	Notation	Description
$\mathcal{U}$	Set of all ground users (UEs)	$\boldsymbol{\varpi}$	Path association vector
$\mathcal{U}_M$	Set of all ground users associated to the main BSs	ϱ	Cell association vector
$\mathcal{U}_N$	Set of all UEs associated to the neighbor BSs	$\bar{T}$	Throughput threshold
$\mathcal{B}_M$	Set of main BSs	S	State vector in the RL approach
$\mathcal{B}_N$	Set of neighbor BSs	A	Action vector in the RL approach
B	Bandwidth	P_r	Transition probability
$u_l = (x_l, y_l, z_l)$	UAV's longitude, latitude, and altitude	γ	Discount factor
V	UAV's pitch speed	$R(s)$	Reward function
$\boldsymbol{p}$	UAV's taken path based on locations, i.e., l_i	w	Reward function's weights
$\mathcal{L}$	Set of all locations s.t. $l_i \in \mathcal{L}$	$\phi(s)$	Features vector
$d_{a,b}$	Distance between locations a and b	$\pi(S)$	Next action policy
$\varsigma_{b,l}$	SNR received at BS b at location l	$V^{\pi}(s)$	Policy value at state s
f	Center frequency	$\mu(s)$	Feature expectation value
$\Phi_{b_m,l}$	Received throughput at BS b_m	D	Hyper distance
$P_{b_m,l}$	UAV's transmission power	$Q(s,a)$	Q-Action value at state s
$h_{b_m,l}$	Channel gain between the UAV and the BS b_m	θ	Stochastic Gradient Descent weights
$G_{b_m,l}$	Antennas gain between the UAV and BS b_m	MSE	Mean Squared Error
N_0	Noise at the receiver	τ	Tuple of states-actions trajectories

training. [43] presented a heuristic algorithm for UAV path planning while maintaining Ground Base Station connectivity. They simplified the problem into a Traveling Salesman Problem (TSP) and employed a combination of A*, Genetic Algorithm (GA), Simulated Annealing Algorithm (SAA), and RL. [44] presented the Hexagonal Area Search (HAS) algorithm integrated with a multi-agent DQN, adeptly addressing collision issues among UAVs.

2.2.3. *DRL-based approaches for joint optimization*

Authors in [61] proposed a DRL-based approach to jointly optimize path planning and jamming power allocation for UAVs aimed at suppressing netted radar systems. Using the Proximal Policy Optimization (PPO) algorithm, the UAV's trajectory and jamming power are dynamically adjusted to balance stealth and radar disruption, significantly improving UAV serviceability and mission success rates. [62] developed a reinforcement learning-assisted framework for multi-UAV task allocation and path planning in IIoT applications. The multi-agent reinforcement learning (MARL) algorithm enables efficient, conflict-free path finding and task distribution, showing superior performance in complex industrial settings. [63] proposed a DRL-based scheme for priority-aware path planning and user scheduling in UAV-mounted MEC networks. The utility maximization problem, which optimizes both the UAV's path and user offloading strategy, is effectively addressed by the DRL-based method, leading to improved system utility. [64] introduced a DRL-based resource allocation and trajectory planning framework in NOMA-enabled multi-UAV collaborative caching systems in 6G networks. This approach learns to optimize resource distribution and UAV paths in real time, significantly improving resource utilization and data throughput. Additionally, [65] proposed a multi-agent DRL approach to optimize UAV trajectory, scheduling, and access control in UAV-assisted wireless sensor networks, improving energy efficiency through hierarchical learning that reduces action and state spaces, enhancing learning stability and sensing performance. Moreover, the authors of [66] utilized PPO and DQN algorithms to jointly optimize UAV positioning and radio spectrum resource allocation, aiming to maximize cumulative discounted rewards and effectively address the non-convex optimization challenges in dynamic UAV wireless networks.

While traditional optimization methods offer precise mathematical solutions, they often lack the flexibility and scalability needed for dynamic environments. Common AI-based approaches, such as RL, depend heavily on well-defined reward functions, which can be challenging to formulate in complex, multi-objective scenarios. Our proposed approach uniquely addresses these challenges by leveraging apprenticeship learning through IRL combined with both Q-learning and DQN. Unlike conventional RL methods that require meticulously crafted reward functions, our method learns the optimal policy from expert-generated experience data, allowing the UAV to replicate complex decision-making behaviors observed in successful missions. This is particularly advantageous in urban air mobility applications, where safety and adaptability are paramount. By balancing multiple objectives such as throughput, communication coverage, energy consumption, and interference, our IRL-based approach ensures robust performance in highly dynamic environments. The assumptions of the related works are summarized in Table 1.

3. System model

Consider a single cellular-connected UAV with different payloads such as sensors, GPS, and cameras. The equipment collects data and sends it to a fusion center for further processing. We consider an orthogonal frequency-division multiple access (OFDMA) scheme for the wireless communication link, which divides the total cell bandwidth B into K resource blocks (RBs). The UAV utilizes an uplink (UL) to transfer data to the ground base station (BS). Fig. 1 depicts the system model. This figure shows that the UAV's operation field is divided into multiple non-overlapping hexagonal cells, each served by one BS, which services several terrestrial UEs with random movement. The UAV can fly over these cells at a fixed height to reach its destination. The BSs are categorized into two groups: (1) the main network to which the UAV is linked, and the BS provides the required bandwidth for the UAV's UL. UAV's transmission does not interfere with the communication of other terrestrial UEs in the same cell; (2) The neighbor BSs that the UAV does not communicate with. The term 'Neighbor BS' encompasses all adjacent base stations that may experience UAV interference. However, the UAV's transmission causes interference to the neighbor BSs and their UEs when the UAV uses the same RB with them. We assumed the neighbors' UEs could not interfere with the UAV's transmission since the transmission power was not strong enough [42]. Hence, the number of terrestrial UEs in neighbor cells is critical because they are affected by the UAV's interference. Here, our design considers the UAV's interference value on individual UEs. The set $\mathcal{U}$ of U ground UEs consists of two groups: $\mathcal{U}_{\mathrm{M}} \subseteq \mathcal{U}$ and $\mathcal{U}_{\mathrm{N}} \subseteq \mathcal{U}$, which are sets of UEs in the main and neighbor networks, respectively.

The base stations in the main and neighbor networks are shown as $b_m \in \mathcal{B}_{\mathrm{M}}$ and $b_n \in \mathcal{B}_{\mathrm{N}}$, respectively. The UAV has a predefined initial source and destination. Its task is to reach the destination while keeping its communication with a legitimate base station, b_m. The UAV and other base stations are defined in a hexagon grid area where the ground BSs are fixed, and the UAV can move in six directions (North, North East, South East, South, South West, North West) at a fixed height. While near the zone's edge, the UAV cannot fly off the grid. For the sake of simplicity, the BS is assumed to be in the center of each hexagon. The UAV's location is known based on the onboard GPS sensor, $u_l =$

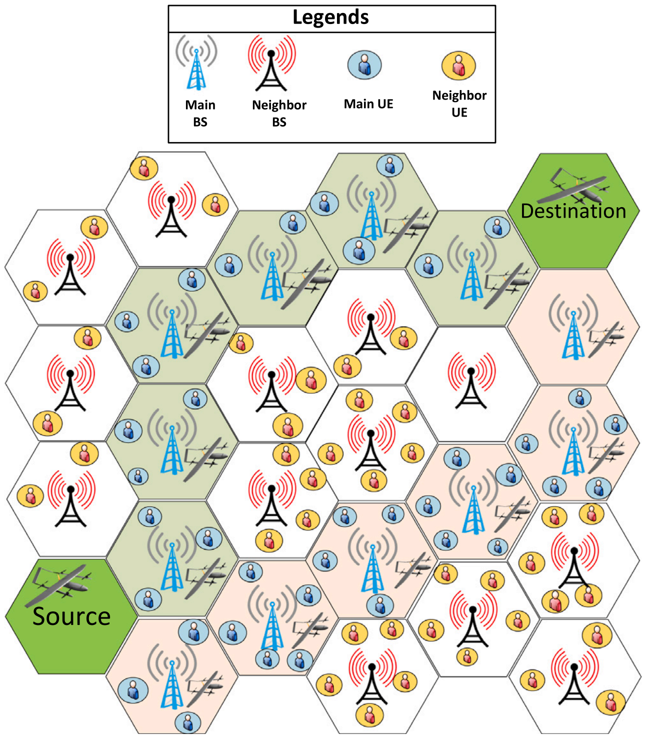


Fig. 1. A sample system model showing two different possible paths of the UAV from the source to destination. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(x_l, y_l, z_l). The UAV changes its location with a constant pitch speed velocity, V, $0 < V \leq \bar{V}$, where $\bar{V}$ is the maximum available pitch speed for the UAV. Also, the UAV is assumed to change its direction using the yaw action, not rolling. The UAV's path from the source to destination is shown as $\boldsymbol{p} = (l_1, l_2, \ldots, l_f)$, which the first location is the source cell, $l_1 = S$ and the last one is the destination cell, $l_f = D$. All locations are subsets of the global location set $\mathcal{L}$, $(l_1, l_2, \ldots, l_L) \in \mathcal{L}$, where L is the grid size or the number of hexagons. The UAV's path length is not fixed, but if it cannot reach the destination in a certain number of movements, the UAV's operation may fail due to battery limitations. Based on the literature [67], the grid area is assumed to be small enough for the UAV's location to be assumed constant inside each hexagon.

3.1. Wireless communication model

Buildings and obstructions impact on signal propagation from the air to the ground. Depending on the environment, this connection may be a line of sight (LoS) or a non-line of sight (NLoS) link. Accordingly, the path loss of LoS and NLoS links of the UAV at location l and BS b_m from the primary network on a sub-6 GHz band can be described as follows [68]:

$$\varsigma^{\mathrm{LoS}}_{b_m,l}\ (\mathrm{dB}) = 20\log_{10}(d_{b_m,l}) + 20\log_{10}(f) - 147.55 + \eta_{\mathrm{LoS}}, \tag{1}$$

$$\varsigma^{\mathrm{NLoS}}_{b_m,l}\ (\mathrm{dB}) = 20\log_{10}(d_{b_m,l}) + 20\log_{10}(f) - 147.55 + \eta_{\mathrm{NLoS}}, \tag{2}$$

where l is the UAV's location or the center of the cell in which the UAV is currently located. f is the center frequency and $d_{b_m,l}$ is the distance between BS b_m and the UAV located at location l. Also, η_{LoS} and η_{NLoS} are LoS and NLoS links attenuation factors, respectively. The probability of having a LoS link depends on environmental factors. In our model, for UAV to the ground link, the LoS probability is given by the proposed model in [68]:

$$\mathcal{P}_{\mathrm{LoS}}(d_{b_m,l}) = \frac{1}{1 + c_1 \exp\left(-c_2\left(\frac{180}{\pi}\tan^{-1}\left(\frac{H}{d_{b_m,l}}\right) - c_1\right)\right)}, \tag{3}$$

where H represents the UAV height, and c_1 and c_2 are environmental dependent permanents. Accordingly, the total path loss of UAV at location l and ground BS b_m can be defined as

$$\varsigma_{b_m,l} = \mathcal{P}_{\mathrm{LoS}}\ \varsigma^{\mathrm{LoS}}_{b_m,l} + \left(1 - \mathcal{P}_{\mathrm{LoS}}\right)\ \varsigma^{\mathrm{NLoS}}_{b_m,l}. \tag{4}$$

The SNR for the UAV UL data transmission to the main BS b_m is written as $\Phi_{b_m,l}$:

$$\Phi_{b_m,l,k} = \frac{P_{b_m,l,k} h_{b_m,l,k}}{N_0}, \tag{5}$$

where $P_{b_m,l,k}$ is the transmission power that the UAV allocates to each RB and sends its data to BS b_m at location l. $h_{b_m,l,k} = G_{b_m,l,k} 10^{-\varsigma_{b_m,l}/10}$ indicates the channel gain between the UAV and the main BS b_m based on the path loss defined in (4), where $G_{b_m,l,k}$ denotes the antenna gain values of the UAV and the BS (b_m). While we assumed a simplified model in this study to calculate the gain value, in the real-world scenario, the antennas of the BSs are down-tilted to have better coverage for the ground UEs and reduce the effect of inter-cell interference. Hence, the UAV is usually served by the side-lobe antenna of the ground BS. N_0 is the noise power spectral density at the main receiver BS b_m. Based on the SNR definition in (5), and by considering B_k as the bandwidth of RB k, the throughput rate for the UAV at location l and based on the connected BS b_m can be written as:

$$T_{b_m,l} = \sum_{k=0}^{K} B_k \log_2(1 + \Phi_{b_m,l,k}). \tag{6}$$

The UAV may interfere with the terrestrial UE transmission in the neighbor networks. This interference depends on the UAV's transmission power, quality of the UAV's channel to the neighbor BS b_n and the probability of allocating the same resource block in the UAV and terrestrial UEs. Due to minimum knowledge sharing, the UAV is blind to the knowledge of UEs in the neighbor networks. Therefore, we assume that at least one UE in neighbor cells is assigned with the same resource block. Accordingly, the UAV's interference in the uplink of a UE $u \in \mathcal{U}_{\mathrm{N}} \subseteq \mathcal{U}$ at BS b_n, and by considering $\mathcal{K}_u$ as a set of allocated RBs to UE u, will be [42]:

$$I^u_{b_n,l} = \sum_{m=1}^{\mathcal{B}_{\mathrm{M}}} \sum_{k=0}^{|\mathcal{K}_u|} P_{b_m,l,k} h_{b_n,l,k}, \tag{7}$$

where $P_{b_m,l,k}$ is the UAV's transmission power in RB $k \in \mathcal{K}_u$ when it was transmitting to the main BS $b_m \in \mathcal{B}_{\mathrm{M}}$ at location l on the same RBs of UE u. Also, $h_{b_n,l,k}$ indicates the channel gain between the UAV and the neighbor BS b_n. This interference rate is an argument that affects the terrestrial UE u SINR and throughput rate:

$$\Phi^u_{b_n} = \frac{P^u_{b_n} h^u_{b_n}}{I^u_{b_n,l} + N_0}, \tag{8}$$

$$T^u_{b_n} = B \log_2(1 + \Phi^u_{b_n}), \tag{9}$$

where $P^u_{b_n}$ is the transmission power for UE u. For the sake of simplicity, all UEs' transmission powers are assumed to be constant. Also, $h^u_{b_n}$ indicates the channel gain between the UE u and the BS b_n.

3.2. Objective definition

Based on the defined system model, the UAV's objective is to fly from the source point and reach the destination point based on the BSs' locations and the density of the UEs in neighbor cells to minimize the interference level it causes on the neighbor terrestrial UEs, maximize its UL throughput by finding an optimal transmission power value, and the shortest path between the source and destination.

The path association vector based on the hexagons is shown as $\boldsymbol{\varpi}$ that consists of elements $\varpi_{a,b} \in \{0, 1\}$. The value 1 means that the UAV has moved from location a to b, and 0 means otherwise. The UAV has to find the optimized path association vector from the source to destination $\boldsymbol{\varpi}^*$. To meet these requirements, the UAV has to find the optimal transmission power $P^*_{b_m,l}$ at each location l where the UAV is associated with tower b_m, which is the main base station. The optimal transmission power $P^*_{b_m,l}$ for transmission to base station b_m can be chosen from the transmission power range $P^*_{b_m,l} \in [P_{\min}, P_{\max}]$ which includes a vector of discrete values. $P^*_{b_m,l}$ should meet the satisfactory throughput rate $T_{b_m,l} > \bar{T}$, where $\bar{T}$ is the minimum acceptable throughput. The optimal power vector during the UAV's flight is shown as $\boldsymbol{P}^*$. To find $P^*_{b_m,l}$, the UAV has to find its BS-connectivity association vector. This connectivity vector, $\varrho_{b_m,l} \in \varrho$, shows whether the UAV connects to BS b_m at location l. The values for the connectivity vector are $\{0, 1\}$.

The UAV has to find the best BS connectivity vector ϱ^* based on other optimization variables in this problem. Since the UAV does not cause any interference on the UEs in its current cell and it can interfere with other neighbor UEs in neighbor cells, the optimal variables such as $\boldsymbol{\varpi}^*$, $\boldsymbol{P}^*$, and ϱ^* are chosen to minimize the interference to neighboring terrestrial UEs. In a nutshell, this optimization problem can be written as:

$$\min_{\boldsymbol{P},\boldsymbol{\varpi},\varrho} \quad \alpha \sum_{l=l_1}^{l_L} \sum_{n=1}^{|\mathcal{B}_N|} I^u_{b_n,l} + \beta \sum_{a=l_1}^{l_L} \sum_{b=l_1}^{l_L} \varpi_{a,b},$$

$$+ \delta \sum_{l=l_1}^{l_L} \sum_{m=1}^{|\mathcal{B}_M|} \varrho_{b_m,l}, \tag{10a}$$

$$\text{s.t.,} \quad P_{\min} \le P_{b_m,l} \le P_{\max}, \; b_m \in \mathcal{B}_M, \tag{10b}$$

$$\varpi_{a,b} \in \{0,1\}, \; \varrho_{b_m,l} \in \{0,1\}, \; l, a, b \in \mathcal{L}, \tag{10c}$$

$$\sum_{l=l_1}^{l_L} T_{b_m,l} \ge \varrho_{b_m,l} \bar{T}, \tag{10d}$$

$$\sum_{m=1}^{|\mathcal{B}_M|} \varrho_{b_m,l} - \sum_{b=l_1, b\neq a}^{l_L} \varpi_{b,a} = 0, \tag{10e}$$

$$\sum_{a=l_1}^{l_L} \varpi_{S,a} = 1, \qquad a \neq S, \tag{10f}$$

$$\sum_{b=l_1}^{l_L} \varpi_{b,D} = 1, \qquad b \neq D, \quad b \neq S, \tag{10g}$$

$$\sum_{a=l_1, a\neq b}^{l_L} \varpi_{a,b} - \sum_{k=l_1, k\neq b}^{l_L} \varpi_{b,k} = 0, \tag{10h}$$

$$\sum_{a=l_1, a\neq b}^{l_L} \varpi_{a,b} \le 1. \tag{10i}$$

where parameters α, β, δ serve as weight coefficients to represent the relative importance of different objectives—uplink interference on UEs, path association vector, and BS connectivity vector, respectively. The optimization problem in (10a) aims to minimize the interference level on neighbor network UEs and BSs; also, it minimizes the UAV's path length from the source to the destination and selects the main BSs for the communication, which imposes less interference to neighbor networks. Eqs. (10b) and (10c) are the optimization problem constraints regarding the UAV's transmission power, the UAV's path between different hexagon cells, and the UAV-BS connectivity vector. Eq. (10d) guarantees that the throughput rate for the UAV-BS UL at each location l when it is associated with BS b_m is greater than a predefined threshold $\bar{T}$, Eq. (10e) guarantees that the UAV is only connected to one main BS b_m at each location l on its path to the destination. Eq. (10f) claims that the UAV's path starts with the source point S and the UAV only visits the source one time, (10g) claims that each UAV's path is ended at the destination D, (10h) guarantees that if the UAV visits any midpoint such as b between the source S and destination D, then it will leave that point and fly to another cell. (10i) claims that each hexagon b on the UAV's path will be visited at most once.

Fig. 1 shows a sample scenario of this system model. The UAV chooses two paths from the source to the destination; the green one is the optimal path considering the density of the terrestrial UEs since the neighbor cells have lower densities, and the UAV's UL communication has less interference on the terrestrial UEs. We should note that the UAV can allocate an optimal value for the transmission power in each cell to control the trade-off between the throughput satisfaction threshold and the level of interference. On the other hand, the orange path shows a route in which the neighbor cells carry a more significant density regarding the number of UEs, and as a result, the UAV faces a more strict restriction on its transmission power.

This optimization problem is categorized as a mixed integer non-linear one, which is hard to solve. This study aims to consider an autonomous approach as a solution for the UAV to find the optimal values for the UAV's transmission power, path, and cellular connectivity vector. This multi-parameter problem involves a huge state-space set, which is overwhelming when solving and finding the optimal values in a reasonable amount of time. To investigate the possible solution for this problem, we utilize an apprenticeship learning approach and use the expert's knowledge to expedite the solving problem and better understand the optimal reward function.

4. Background on IRL

Many artificial intelligence and robotic applications consist of autonomous agents such as ground robots, UAVs, and self-driving cars. These agents usually operate in an uncertain and complex environment where decision-making is cumbersome and necessary.

In multi-objective reinforcement learning problems, where the agent deals with various and possibly conflicting objective functions to perform tasks of different natures, it is not always possible to define a reward function that optimally captures all the objectives (10a)–(10i). The problem studied in this paper also involves multiple actions, such as movement and level of transmission power, with different natures and behaviors. This problem makes it more intense for the agent to understand the reward function meaningfully. Usually, in these situations, an expert with enough experience in the domain and field can demonstrate appropriate behavior to the agent. These demonstrations may save some training time while assisting the agent in determining the best reward function based on the most optimal presented behavior. This type of learning is referred to as Learning from Demonstration (LfD) or Programming by Demonstration (PbD), and it is also known as imitation learning (IL). IL techniques are categorized into two branches: (i) behavioral cloning [69] and (ii) apprenticeship learning via IRL [70,71].

In BC, the agent blindly mimics the expert or teacher without considering the expert's optimality, which means that if the expert makes a wrong decision, the expert may follow that. Usually, IL problems are solved using supervised learning approaches. However, if the expert has yet to experience the state the agent observes, the learned policy may not achieve an optimal outcome. This issue can be solved by an approach called Data Aggregation (DAGGER) [72]. DAGGER aims to collect expert demonstrations based on the state distribution observed by the agent. In this case, the expert should always be available, which is impossible in all scenarios and applications.

On the other hand, apprenticeship learning via IRL tries to find the expert's hidden desire (reward function) from the demonstrations to find the optimal policy [51,73]. This paper utilizes apprenticeship learning via IRL to solve the problem proposed in Section 3. Similar approaches such as BC, Q-Learning, and Deep Q-Network are also implemented to evaluate the numerical performance.

5. UAV's self-organizing approach

5.1. Apprenticeship learning via inverse reinforcement learning

The problem definition of this study can be shown as a sequential decision-making process that can be posed in the (finite-state)

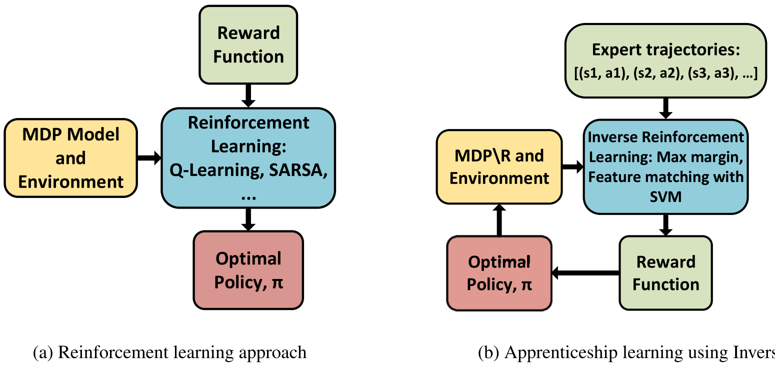

Reward Function
MDP Model and Environment
Reinforcement Learning: Q-Learning, SARSA, ...
Optimal Policy, π
(a) Reinforcement learning approach
Expert trajectories: [(s1, a1), (s2, a2), (s3, a3), ...]
MDP\R and Environment
Inverse Reinforcement Learning: Max margin, Feature matching with SVM
Optimal Policy, π
Reward Function
(b) Apprenticeship learning using Inverse RL

Fig. 2. Difference between reinforcement learning and Inverse RL for apprenticeship learning (SARSA: State–action–reward–state–action, SVM: support vector machine).

MDP. The reward function is typically provided in MDP problems, allowing the agent to determine the optimal transition probabilities, value function, and policy. The agent can utilize the optimal policy for decision-making in every state. However, assuming that the reward function is present in every complex system model is overly optimistic, and manually obtaining the reward function is challenging.

The existence of an expert may make it possible to have a set of trajectories and demonstrations to learn the optimal behavior. The expert could be a virtual agent that operates on a high-performance computing machine but cannot be utilized onboard a UAV. In this study, the term "*trajectory*" refers to a set of states and actions rather than the UAV's trajectory movement or path. The IRL aims to re-construct the unknown reward function $R(\tau)$ from the expert demonstrations. Later, this obtained reward function can be used in an RL approach to find the optimal policy. However, this problem is "*ill-posed*" since multiple reward functions exist for a unique and optimal policy.

Here, we assumed that the expert is trying to optimize an unknown reward function, which this unknown function can be represented as a linear mixture of known "*features*". Fig. 2 shows the main difference between a reinforcement learning problem and an apprenticeship learning via IRL (AL-IRL) problem. The reward function is available in Fig. 2(a), and the optimal policy is gained based on that. However, in Fig. 2(b), the reward function is found based on the IRL and the expert data, and the optimal policy is obtained. Expert trajectories are not the same as optimal policies; however, they describe optimal behavior.

We assumed that the problem is MDP\R, which shows the MDP problem without a reward function. This MDP\R consists of a tuple of $\langle \mathcal{S}, \mathcal{A}, Pr, \gamma, T \rangle$, where $\mathcal{S}$, $\mathcal{A}$, and Pr represent the state space, action space, and transition probability from current state $s \in \mathcal{S}$ to the next state $s' \in \mathcal{S}$, respectively. Also, γ is a discount factor, and T is a set of trajectories demonstrated by an expert. The MDP\R tuples are described as follows:

(1) State

To define the state $s \in \mathcal{S}$, we assume that a vector of features $\boldsymbol{\phi} : s \rightarrow [0, 1]^k$, where k is the number of features. $\mathcal{S}$ represents a finite set of feature states, $\mathcal{S} = \{\phi_i | 1 \leq i \leq 5\}$. Each state $s \in \mathcal{S}$ is displayed with five feature functions.

- $\phi_1(s)$ is the feature that keeps track of the UAV's distance to the destination. This feature helps the agent to have a better understanding of the shortest path.
- $\phi_2(s)$ is the feature that keeps track of the UAV's hop counts as its traveled distance. This feature helps the agent to perform its task in a limited number of hop counts because of UAV's battery life constraint.
- $\phi_3(s)$ keeps track of the UAV's successful task, where the term "successful" refers to the cases in which the UAV reaches the destination cell. $\phi_3(s)$ behaves like a binary variable, where it takes the value of "0" if the UAV does not reach the destination and "1" for reaching the destination cell.
- $\phi_4(s)$ is a feature for the amount of received throughput from the UAV to the designated BS based on the allocated transmission power and SNR defined in (5) and (6) at each state.
- $\phi_5(s)$ keeps the value for the interference imposed on the neighbor UEs in the neighbor cells.

All feature values are normalized to the range of [0, 1]. This condition on the reward function is necessary for the IRL algorithm to converge. Hence, each state can be expressed based on the feature value functions as,

$$\boldsymbol{\phi}(s_t) = \{\phi_i(s_t), 1 \leq i \leq 5\}. \tag{11}$$

(2) Action

$\mathcal{A}$ represents the action space, which consists of the UAV's movement action and the transmission power allocation $\boldsymbol{P}$. The UAV's movement action somehow includes the defined movement variables $\boldsymbol{\varpi}, \boldsymbol{\varrho}$ in (10a). The UAV chooses both actions autonomously at the same time.

(3) Reward

Here, the unknown reward function is defined by considering the feature functions in the state space. The reward function is defined as:

$$R(s) = \boldsymbol{w}^* \cdot \ \boldsymbol{\phi}(s), \tag{12}$$

where, $\boldsymbol{w}^*$ are the optimal weights for the features to define the desirable optimal reward function, $\boldsymbol{w} \in \mathbb{R}^k$. To have the convergence for the IRL, it is necessary to bound the reward function by -1 and 1. Therefore, the weights are also bounded $\|\boldsymbol{w}^*\| \leq 1$. Based on the definition in (12) and the defined features, the reward function can be re-written as:

$$\begin{aligned} R(s, a, s') &= \boldsymbol{w}^T \cdot \ \boldsymbol{\phi}(s_t), \\ &= \sum_{i=1}^{5} w_i \phi_i(s, a, s'). \end{aligned} \tag{13}$$

In IRL approaches, researchers have the flexibility to define reward functions in two main categories: linear and non-linear functions. The choice between them depends on the problem's nature and the environment's characteristics. In this study, the reward function is specifically introduced as a linear combination of feature values, $\boldsymbol{\phi}(s_t)$, as described in (13). A linear reward function offers enhanced interpretability, where the weight of each feature in the linear combination directly reflects its significance in shaping the agent's behavior. This transparency proves beneficial for a comprehensive understanding of the learned policy. Furthermore, the inherently linear nature of this function contributes to computational efficiency, often leading to quicker convergence during the training process. Importantly, it is less susceptible to overfitting, facilitating generalization across diverse scenarios. The pragmatic choice of using a linear function is particularly apt in scenarios where such a representation effectively captures the essential relationships between features and rewards. These features make linear functions a commonly adopted form for reward functions in IRL methods, often represented as a linearly weighted combination of reward features [51,52].

It is worth mentioning that not all feature value functions behave positively for the reward function. For instance, throughput is encouragement, and interference is punishment. The IRL algorithm aims to find the optimum value for the weight vector, $\boldsymbol{w}^*$, to express the expert's behavior for this specific problem. Let us define π as a policy that maps the visited states to probabilities on the action vectors for the decision-making process. The value of each policy π can be defined as an expected value of the summation of discounted reward values based on the chosen policy:

$$\begin{aligned}\mathbb{E}\left[V^{\pi}(s)\right] &= \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty}\gamma^{i}R(s_i,a_i,s_i')\right],\\ &= \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty}\gamma^{i}\ \boldsymbol{w}\ .\ \boldsymbol{\phi}(s_i,a_i,s_i')\right],\\ &= \boldsymbol{w}\ \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty}\gamma^{i}\boldsymbol{\phi}(s_i,a_i,s_i')\right],\\ &= \boldsymbol{w}\ \mu(\pi),\end{aligned} \tag{14}$$

where i is the step the agent or the UAV takes in every episode. $\mu(\pi)$ is defined as a feature expectation value for the policy π, $\mu(\pi)\in\mathbb{R}^k$, and k is the number of defined features.

To find the unknown parameters, $\boldsymbol{w}$, in the IRL algorithm, different tools such as feature expectation matching [51,70], maximum margin planning [74,75], and maximum causal entropy [76] have been proposed. This paper uses the feature expectation matching between the expert and the learner UAV. The reason is that the behavior of the agent and expert can be expressed based on different explicit features; hence, feature expectation matching can be used in this scenario.

We assume that an expert exists who has access to the simulator environment. The expert has a complete understanding and knowledge of the problem and scenario, which means it considers the UEs' density, shortest path, throughput, and interference for the sake of the optimization problem defined in (10a)–(10i). The expert simulates a finite number of trajectories to show the desired behavior. These expert behaviors can be saved in terms of trajectories as some vector of features state $\boldsymbol{\phi}(s)$ and actions $a(s)\in\mathcal{A}$. Later, these trajectories can be used to define the expert feature expectation value by getting an average over several trajectories as follows:

$$\bar{\mu}_E = \frac{1}{\mathrm{N}}\sum_{j=1}^{\mathrm{N}}\ \sum_{t=0}^{\infty}\gamma^{t}\phi(s_t^j), \tag{15}$$

where N is the number of trajectories and j denotes the index for each trajectory. This μ_E or $\mu(\pi_E)$ carries the expert policy for the demonstrated trajectories.

Claim. *The goal is to find a policy π^* such that the second-norm distance between the expert feature expectation and the current policy feature expectation is less than a threshold ϵ_{IRL}. The mentioned policy will meet the criteria for the value of that policy as well which means the first-norm distance between the two value functions for the expert policy and learned policy based on the unknown reward function is less than the same threshold ϵ_{IRL}.*

Proof. The first-norm distance between the expert's and agent's value function is written as,

$$\begin{aligned}\mathrm{D} &= \left|\mathbb{E}\left[V^{\pi_{\mathrm{E}}}\right]-\mathbb{E}\left[V^{\pi^*}\right]\right|,\\ &= \left|\mathbb{E}_{\pi_{\mathrm{E}}}\left[\sum_{i=0}^{\infty}\gamma^{i}R(s_i,a_i,s_i')\right]-\mathbb{E}_{\pi^*}\left[\sum_{i=0}^{\infty}\gamma^{i}R(s_i,a_i,s_i')\right]\right|,\\ &= \left|\boldsymbol{w}^T\mu(\pi^*)-\boldsymbol{w}^T\bar{\mu}(\pi_{\mathrm{E}})\right|,\\ &\le \|\boldsymbol{w}\|_2\ \left\|\mu(\pi^*)-\bar{\mu}(\pi_{\mathrm{E}})\right\|_2,\\ &\le 1\ .\ \epsilon_{\mathrm{IRL}},\\ &\le \epsilon_{\mathrm{IRL}}.\end{aligned} \tag{16}$$

This shows that finding the optimal weights for the reward function such that $\|\mu(\pi^*)-\bar{\mu}(\pi_E)\|_2\le\epsilon_{\mathrm{IRL}}$ guarantees that the distance between the value function for the expert and the optimal policy is less than the same threshold. □

5.2. Support vector machine problem formulation

To find the weights for the reward function, the authors of [51] defined the problem as a support vector machine (SVM) problem, where the aim is to maximize the difference between the value function for the expert optimal policy and other previous learned policies. In the SVM problem, this maximization problem is mapped to maximize the distance between hyperplanes. This SVM problem aims to find a hyper-distance or margin (D) such that the difference between the expert value function and all other previously learned policies is D. This SVM problem can be written as

$$\left|\boldsymbol{w}^{*^T}\mu(\pi_i)-\boldsymbol{w}^{*^T}\bar{\mu}(\pi_E)\right|\le\Delta, \tag{17}$$

$$\boldsymbol{w}^{*^T}\left|\mu(\pi_i)-\bar{\mu}(\pi_E)\right|\le\Delta, \tag{18}$$

where, i is index of the ith learned policy using the reinforcement learning approach. To solve this SVM problem, the class for the expert policy is labeled as "+1", and all other learned policies from the RL tool are labeled as "−1". Based on the defined SVM problem, the distance between the origin and the expert policy class is $\frac{1}{\|\boldsymbol{w}\|_2}$ and the distance between the origin and all other learned policies is $\frac{-1}{\|\boldsymbol{w}\|_2}$, hence, maximizing the distance between hyper-planes is equal to minimizing $\|\boldsymbol{w}\|_2^2$. As a result, (17) can be written as:

$$\min_{\boldsymbol{w}}\quad \|\boldsymbol{w}\|_2^2\ , \tag{19a}$$

$$s.t.\quad \boldsymbol{w}^T\mu_E\ge 1, \tag{19b}$$

$$\boldsymbol{w}^T\mu_{\pi_i}\le -1\ . \tag{19c}$$

Since this SVM problem looks like a quadratic programming (QP) problem, any convex optimization tools or QP solvers such as CVXPY [77] or CVXOPT [78] can solve this optimization problem. Here, CVXOPT is used in Python to find the weights. The objective function of this optimization, (19a), is to minimize the weights and the variables are the weights as well. Constraint (19b) is the subject for the expert policy with the label of "+1" and constraint (19c) is the subject for all the learned policies at different iterations with the label of "−1". It is worth mentioning that the algorithm of finding the optimal weight such that the distance is less than a threshold ϵ_{IRL} is iterative, and the number of learned policies is increasing in the QP. Hence, the optimality of the given solution is crucial to solving this problem. The standard format of any QP problem looks like the following,

$$\min_{x}\quad \frac{1}{2}x^TPx+q^Tx, \tag{20a}$$

$$s.t.\quad Gx\le h, \tag{20b}$$

$$Ax=b\ . \tag{20c}$$

To match this with our SVM problem (19a), matrices of q, A, and b are all zeros and matrix P is $2\mathbf{I}_K$, which $\mathbf{I}$ is the identity matrix and K is the number of features (K = 5 in our problem). The dimension of matrices G and h changes and increases with each iteration. Then, in nth iteration for different weights and learned policies, the formats of G and h are

$$G_{(n+1,k)}=\begin{bmatrix}-\mu_E(1) & -\mu_E(2) & \dots & -\mu_E(k)\\ \mu_{\pi_1}(1) & \mu_{\pi_1}(2) & \dots & \mu_{\pi_1}(k)\\ \vdots & \vdots & \vdots & \\ \mu_{\pi_n}(1) & \mu_{\pi_n}(2) & \dots & \mu_{\pi_n}(k)\end{bmatrix}, \tag{21}$$

$$h_{(n+1,1)} = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}, \tag{22}$$

where $\mu_{\pi_n}(k)$ is the kth UAV's feature expectation based on the nth learned policy. These matrices of P, q, G, h, A, and b map the QP problem to our scenario defined in (19a), (19b), and (19c). Algorithm 1 summarizes the approach of the apprenticeship learning via IRL inside the loop iteration to find the optimal weights.

Algorithm 1: Apprenticeship learning via IRL algorithm

```
1  Initialization:
2  Load the simulation environment
3  Load the expert trajectories
4  Calculate the expert feature expectation $\bar{\mu}_E = \frac{1}{N}\sum_{j=1}^{N}\sum_{t=0}^{\infty}\gamma^t\phi(s_t^j)$;
5  Add some random values to the agent feature expectation for
     the first round of optimization
6  while True do
7      Weights = SolveQP(Expert feature exp, Agent feature exp
         list);
8      Weights($w_i$) = Weights / norm;
9      TrainedPolicy($\pi_i$) = learner(Weights);
10     Reset the simulation environment;
11     $\mu_i(\pi_i, w_i)$ = RunSimulation($\pi_i$);
12     Add $\mu_i(\pi_i, w_i)$ to the agent feature expectation list;
13     Hyper distance = D = $|w^T\mu(\pi_i) - w^T\bar{\mu}(\pi_E)|$;
14     if $D < \epsilon_{IRL}$ then
15         Break;
16     end
17     $i$+ = 1;
18 end
19 Return Weight($w_i$), $\mu_i$, $\pi_i$;
```

The main loop of Algorithm 1 continues until it reaches a point where the hyperdistance is less than a threshold and in that case, it breaks from the main loop and returns the optimal weights (w_i), feature expectation (μ_i), and the learned policy (π_i).

To train a model for the optimized weights of the reward function and find the optimal policy to evaluate the hyperdistance, learning methods like RL, especially a temporal difference (TD) learning tool, must be utilized. These methods find the optimal policy on line 9 of the Algorithm 1. In this paper, we use two TD methods: the first one is Q-learning based on linear function approximation, and the second one is DRL or DQN. The following two sections propose these two tools for the mentioned problem of apprenticeship learning to learn the policy.

5.3. AL-IRL with Q-learning using linear function approximation

After finding the normalized weights in apprenticeship learning, the reward function is known for the specific iteration and the problem of MDP\R is converted to a MDP problem with the tuple of $\langle \mathcal{S}, \mathcal{A}, Pr, R, \gamma \rangle$ where R is the reward function defined based on the obtained weights, (12) and (13). The learning policy aims to find a policy that can decide on an action for the agent at each observed state to maximize the immediate reward while also considering the future reward. Different value iteration approaches are available to find this policy [79,80]. We chose the Q-learning tool to solve this problem [81,82]. In the standard Q-learning, the updating rule for the Q values is defined based on

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a')], \tag{23}$$

where $s \in \mathcal{S}$ is the current state, $a \in \mathcal{A}$ is the action chosen based on the policy or random exploration which changes the agent's state from s to s', α is the learning rate, $R(s,a,s')$ is the reward function defined based on the IRL approach from the previous section. γ is the discount factor considering the future reward (value-iteration or $Q(s')$). After updating the Q-values based on different approaches, such as temporal difference (TD) or Monte Carlo (MC), the agent follows the policy rule for exploiting greedy behavior. Later, this policy is considered a learned policy for apprenticeship learning.

$$\pi(s) = \operatorname*{argmax}_a Q(s,a) \ . \tag{24}$$

It is worth mentioning that this Q-updating approach works for problems with discrete states and actions, such as the grid-world cells or scenarios. However, the state in this paper is based on the vector of features $\boldsymbol{\phi}(s)$ based on (11). Because the states are defined as continuous values from the feature vectors, the Q-learning updating rule is more challenging to update. The solution to this problem is to find the Q-value from the defined feature vectors using a linear function approximation (LFA) tool such as linear regression. This means each function of $Q(s,a)$ can be defined based on the values of features:

$$\begin{aligned} Q(s_t,a_t) &= \theta_0 + \theta_1\ \phi_1(s,a) + \theta_2\ \phi_2(s,a) + \cdots \\ &\quad + \theta_\kappa\ \phi_\kappa(s,a) = \boldsymbol{\theta}^T \phi(s,a), \end{aligned} \tag{25}$$

where κ is the number of features, and in this paper, five features are for each visited state. θ_i are the weight vectors to find the value of $Q(s,a)$. To find the $\boldsymbol{\theta}$ vectors, we need to use a batch of sample data to update the values. To solve this linear regression problem, the stochastic gradient descent (SGD) approach minimizes the loss function for the difference between the predicted Q-value and the actual value. The true target value for the Q is obtained based on the Q-value updating rule, and this value is called $Q^+(s,a)$:

$$Q^+(s,a) \approx R(s,a,s') + \gamma \max_{a'} Q(s',a') \ . \tag{26}$$

This Q target is used in the loss function for the SGD:

$$\begin{aligned} L(\theta^{(i)}) &= \frac{1}{2}(Q^+(s,a) - Q(s,a))^2 \\ &= \frac{1}{2}(Q^+(s,a) - \phi(s,a)^T\theta^{(i)})^2 \ . \end{aligned} \tag{27}$$

Now, considering the mentioned loss function, the values of the weights update based on the gradient:

$$\begin{aligned} \theta^{(i+1)} &= \theta^{(i)} - \alpha_S \partial(L(\theta^{(i)}))/\partial\theta \\ &= \theta^{(i)} - \alpha_S \partial(\frac{1}{2}(Q^+(s,a) - \phi(s,a)^T\theta^{(i)})^2)/\partial\theta \\ &= \theta^{(i)} - \alpha_S[Q^+(s,a) - \phi(s,a)^T\theta^{(i)}] \ . \ \phi(s,a) \ . \end{aligned} \tag{28}$$

The optimal values of the weights are obtained based on different episodes, visits, and agent exploration. The agent's exploration and exploitation process is based on the ϵ-greedy process with ϵ decay over the episodes. It means that the agent behaves randomly in the early episodes, and then acts more greedy based on exploitation of Q-values. At the end of this process, the learned policy (π_i) will be used in the apprenticeship learning to get the latest feature expectation values (μ_i). Algorithm 2 summarizes the procedure for policy learning using Q-learning via linear function approximation for a specific weight of the reward function. Algorithm 2 finds and returns the learned policy for the specific reward function. DIST_LIM indicate the UAV's hop count limitation based on the UAV's battery life and flight time limitation is adopted. In this case, the learned policy is the trained SGD model. The size of the SGD model depends on the action vector. For instance, in this paper, the agent has 6 (Mobility) × 6 (Power Allocation) actions to choose from. Hence, 36 SGD models are the learned policy for this specific configuration. In the next section, we propose a deep Q-network instead of the stochastic gradient descent approach for linear function approximation.

Algorithm 2: Q-learning algorithm with Linear Function Approximation

```
Initialization:
Load the simulation environment;
$\epsilon = 1$;
while episode < NUM_EPS do
    $distance = 0$;
    Reset UAV();
    $Done$ = FALSE;
    while distance < DIST_LIM & not DONE do
        if random < $\epsilon$ then
            $action$ = random Action;
        else
            $action$ = GetGreedyAction(SGD Model);
        end
        Update $Location$ ;
        Calculate SNR, Interference, Throughput;
        Calculate Features $(\phi_1(s,a,s'),\dots,\phi_5(s,a,s'))$;
        Calculate immediate reward $R(s) = \boldsymbol{w}^T.\boldsymbol{\phi}(s')$;
        $Q(s')$ = predict(SGD Model, $\phi(s')$);
        if New location is DESTINATION then
            $Done$ = TRUE;
            $Q^+(s,a) = R(s)$;
        else
            hon $Q^+(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$;
        end
        Update SGD Model(SGD Model, $\boldsymbol{\phi}(s)$, $Q^+(s,a)$);
        $distance += 1$;
    end
    if $\epsilon$ > 0.1 & episode > NUM_EPS/10 then
        $\epsilon$ -= 1/NUM_EPS ;
    end
    $episode += 1$;
end
Return SGD Model;
```

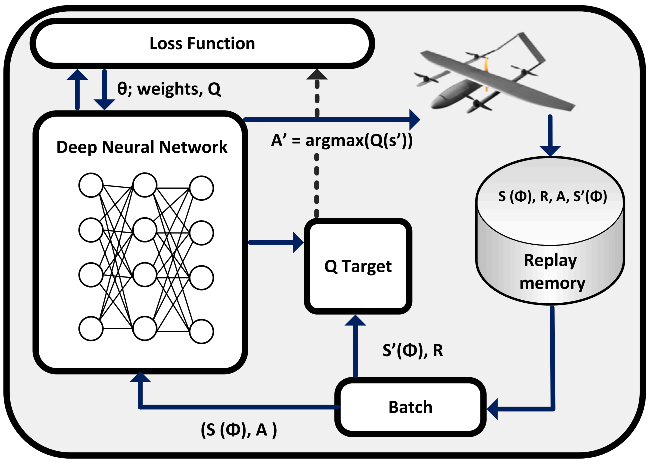


Fig. 3. Deep Q-Network approach for the optimal policy in the apprenticeship learning.

5.4. AL-IRL with deep Q-network

In the previous section, the solution to find the optimal policy was based on the Q-learning algorithm using the linear function approximation with stochastic gradient descent. This section uses the DQN approach to predict the Q-value based on the current features vector (current state) and the chosen action. Fig. 3 demonstrates the deep reinforcement learning structure for the apprenticeship learning for the obtained reward weights based on the inverse reinforcement learning algorithm.

The optimal policy learning block consists of a deep neural network with two hidden layers: an input layer with the size of the number of features for each state and an output layer with the size of all actions. The numbers of neurons in the hidden layers are 30 and 30, respectively. Both hidden layers utilize the Rectified Linear Unit (RELU) [83] for the activation layer. The last layer has the linear activation function since the output of the deep neural networks is a continuous value for the Q-Action-Value. The structure of this DNN is demonstrated in Fig. 4.

The training concept for the exploration and exploitation is similar to Algorithm 2. However, the training process is different, and in this approach, the training model is based on batch samples. The batch samples are picked randomly from the replay memory. Also, a buffer length is considered for the replay memory to keep the data fresh and pop the old data from the replay. Algorithm 3 presents the random batch sample training and fitting those data for the model. This algorithm is summarized since some structures appeared in Algorithm 2. Hence, a few lines are just removed.

Algorithm 3: Deep Q-Network to predict the Q-Action-Value on batch samples

```
while episode < NUM_EPS do
    ...;
    while distance < DIST_LIM & not DONE do
        ...;
        Collect features, actions, reward for the Replay memory;
        if episode > NUM_EPS/10 then
            batch = random.sample(replay, BATCH_SIZE) ;
            for all data in batch do
                x_train = batch[current feature];
                if New location is DESTINATION then
                    $Q^+(s,a) = R(s)$;
                else
                    $Q^+(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$;
                end
                y_train = batch[$Q^+(s,a)$];
            end
            model_DQN.fit(x_train, y_train);
        end
        distance+ = 1;
    end
    if $\epsilon$ > 0.1 & episode > NUM_EPS/10 then
        $\epsilon$ -= 1/NUM_EPS;
    end
    $episode += 1$;
end
Return DQN Model;
```

To train the model and find the optimal weights for the neurons, the Mean Squared Error (MSE) [84] is used as the loss function for the optimization. To perform the stochastic optimization on the MSE, an optimizer called "Adam" is used to solve the problem for each batch of collected data [85]. The MSE loss function for a batch with n samples is shown below

$$MSE = \frac{\sum_{i=1}^{n}(Q^+(s,a) - Q(s,a))^2}{n}, \tag{29}$$

where $Q(s,a)$ is the predicted value based on the current features vector and $Q^+(s,a)$ is the actual values (target value) for the Q-value updated

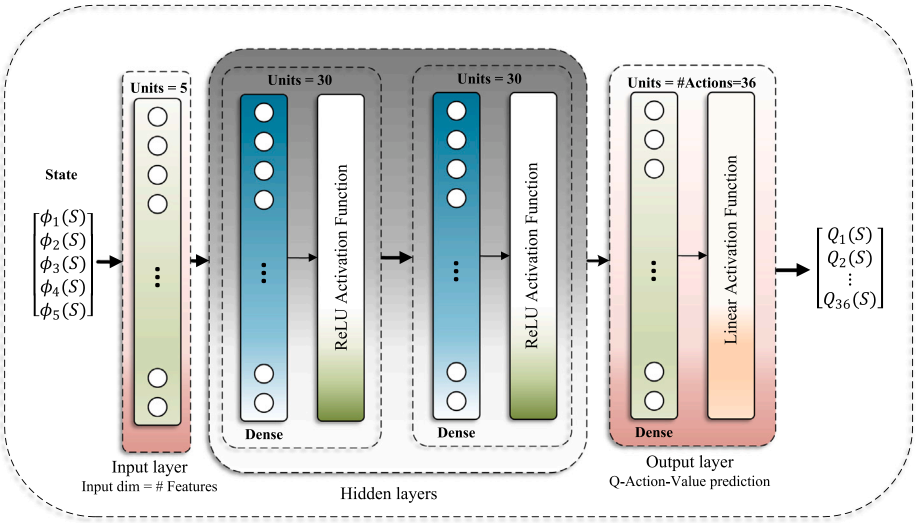


Fig. 4. Deep neural network structure used in the DQN approach to predict the Q-Action value based on the features vector.

based on (26). n is the number of samples in each batch samples for the training.

5.5. Imitation learning: Behavioral cloning

This section proposes the model used in this paper for the BC. Usually, supervised learning tools can obtain the model for imitation learning to mimic the expert's behavior [86]. However, the BC approach only has the solution for those states the expert observes. If the agent or UAV examines a state that the expert has never seen, the outcome may not be optimal. All trajectories from the expert are gathered for imitation learning and BC. The same feature vectors of states with the respective actions are used to train a deep neural network for task classification based on the observed state by the UAV. This problem is categorized as a task classification problem since the UAV wants to choose the right action at the right time based on the expert's experience.

We first collect the expert's data and trajectories to implement the supervised learning for the BC. These trajectories provide a simulated dataset for the training and test evaluation. BC does not need constant access to the expert; however, DAGGER requires the expert's presence in states that have never been visited. Implementing the DAGGER approach is more costly and not always feasible; thus, the BC is applied here for imitation learning. We should note that the collected dataset is based on states and actions, and there is no meaning of the reward function, and the agent mimics the expert blindly.

The expert data, ($\mathcal{D}$), is collected in terms of $\mathcal{D} = \{(\mathbf{x}_t, \mathbf{s}_t, \mathbf{a}_t)\}$, where $\mathbf{x}_t$ is the initial condition for the expert. This paper assumes that $\mathbf{x}_t$ is the location where the agent or UAV starts its initial location $(0,\ 0)$. $\mathbf{s}_t$ is the UAV's state that consists of the features vector defined in (11) in Section 5.1. Action ($\mathbf{a}_t$) is the mixed action based on the mobility and the transmission power allocation, $\mathbf{a} \in \mathcal{A}$. The action space is $\mathcal{A} = \{an(i) | 0 \leq I \leq 35\}$, which consists of 6 movement actions and six transmission powers. Since the agent decides at the same time for both the mobility and transmission power, there are 36 actions. If the actions were independent, the number of actions would be 12 instead of 36. The collected expert trajectories can be shown as,

$$\tau = \{\boldsymbol{\phi}(\mathbf{s})^{(0)}, \mathbf{a}^{(0)}, \boldsymbol{\phi}(\mathbf{s})^{(1)}, \mathbf{a}^{(1)}, \dots\}\ . \tag{30}$$

After collecting the expert trajectories, the problem is defined as multi-class classification and a supervised learning technique is utilized to handle the problem. The supervised learning approach generates a model predicting the appropriate class (action) based on the visited state. A decision tree algorithm is chosen as a supervised learning technique. Next, based on the expert's collected information, the UAV will use this model to decide on the next cell and the next transmission power. Here, the learned model is called π_{BC}

$$\mathbf{a} = \pi_{\text{BC}}(\mathbf{x}_t, \mathbf{s}_t), \tag{31}$$

which BC stands for the BC. To train the model, the decision tree from the SciKit-Learn Python package is used to solve the problem [87,88]. Gini impurity is used to determine the quality of split for various features. After training, the decision tree model has a depth of five with seven leaves.

To illustrate the interaction between the components of our proposed solution, we have provided a sequence diagram in Fig. 5. This diagram shows the interaction between key components: Apprenticeship Learning via IRL in Algorithm 1, SVM, and Q-learning and DQN in Algorithms 2 and 3, respectively. It integrates the block diagrams and structural details of our approach, highlighting the process flow and component interactions. While the diagram specifically illustrates the DQN component, it should be noted that the Q-learning process follows a similar sequence, and thus has not been separately depicted to avoid repetition. Additionally, the overhead and time complexity of the proposed solution are important considerations for practical applications. Detailed analysis of the time complexity for each algorithm is provided in Appendix.

6. Numerical results and experiments

In this section, the designed simulator is explained first, and then the convergence of the apprenticeship learning for the IRL and the policy learner using both the Q-learning and deep Q-network methods is analyzed. Afterward, the performance comparison among various decision-making approaches at the UAV, including inverse reinforcement learning (Q-learning using linear function approximation), inverse Deep Q-Network, BC, shortest path, and a random policy, is utilized to compare the proposed method with similar approaches. Through all the experimental results, a probabilistic channel according to (3) for UAV is considered. However, in some cases, to show the performance comparison results in different UAV channel conditions, we mentioned two different UAV channel scenarios, probabilistic and LoS channels. All designs and simulations are tested on a Ubuntu system with a Ryzen nine 3900X CPU and Nvidia RTX 2080 Ti GPU. The training phase for the deep neural network approach was done based on Tensorflow 2.3.0, Keras 2.4.0 API, CUDA 10.1, and cuDNN 7.6 with Python 3.6.

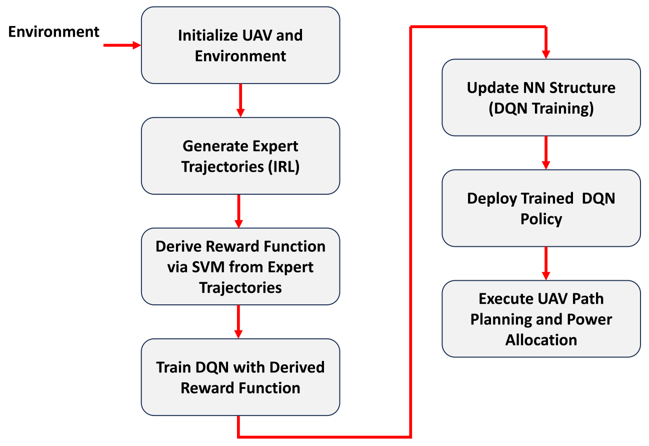

Environment
Initialize UAV and Environment
Generate Expert Trajectories (IRL)
Derive Reward Function via SVM from Expert Trajectories
Train DQN with Derived Reward Function
Update NN Structure (DQN Training)
Deploy Trained DQN Policy
Execute UAV Path Planning and Power Allocation

Fig. 5. Sequence diagram of the proposed AL-IRL via DQN approach.

6.1. Designed simulation environment

In this study, we develop an Open Source simulation environment based on the proposed problem in Section 3. The simulation environment is designed in Python 3.6 to consider the UAV in a pre-defined area with 25 cellular base stations and multiple terrestrial UEs. The UEs' distribution is such that some cells and areas have a higher density, and others are less crowded. While the distribution of UEs is considered fixed, the dynamicity of the environment is maintained through the variability in UEs' power and their connection channels, effectively characterizing the dynamic nature of the environment.

To provide a comprehensive simulation model, we considered key factors relevant to UAV-based communication environments. One common method for modeling air-to-ground propagation channels is to consider LoS and NLoS components along with their occurrence probabilities separately, as shown in [9,89]. In NLoS connections, due to the shadowing effect and reflection from obstacles, the path loss is higher than in LoS connections. We have incorporated these effects into our path loss model, as detailed in Eqs. (1) and (2). Our simulation utilizes a probabilistic Gaussian channel model that includes path loss modeling for both LoS and NLoS scenarios, realistically capturing variations in signal propagation. This allows us to simulate various multipath effects and environmental factors impacting signal strength and quality. Specifically, our model accounts for shadowing from obstacles such as trees and buildings, ensuring realistic representation of signal attenuation. Additionally, the fixed altitude assumption for UAVs reflects practical utilization strategies, where predefined air corridors at specific heights are followed for safe and efficient airspace management. This assumption is supported by relevant literature [41,42,90]. Overall, our approach aligns with established standards and previous literature, providing a robust and realistic representation of UAV communication scenarios.

The proposed solution should be able to find the optimal path regarding the distance, throughput, number of adjacent UEs, and interference on UEs in a dense urban environment with parameters $c_1 = 12.076$ and $c_2 = 0.114$. This simulation is event-based, and the event is the UAV's action. The UAV has six actions for mobility and six discrete actions for transmission power, which results in 36 actions together. The user can change the transmission power range to consider a broader or narrower range to increase or decrease the number of actions. More actions bring more complexity to convergence. This simulator is designed based on an object-oriented approach. Different classes are defined for the UAV, UEs, and BSs. All parameters for the learning algorithms and environment are configurable in the simulator. Table 3 summarizes the main simulation parameters.

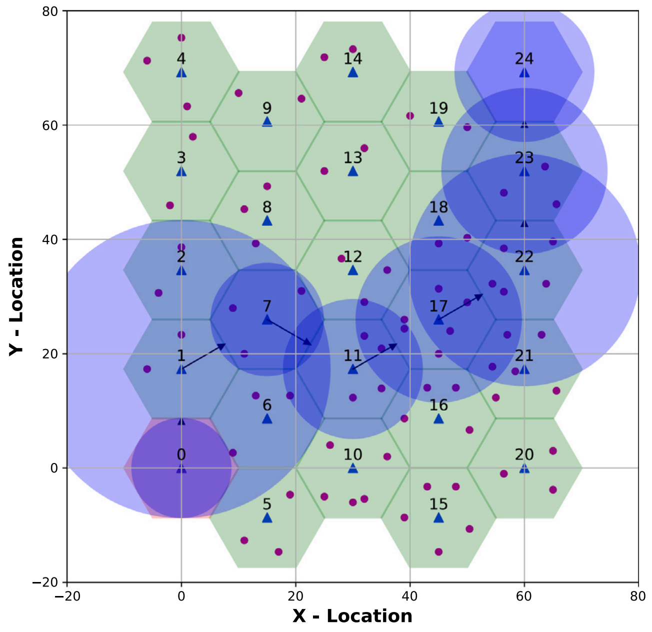


Fig. 6. Demonstration of the designed graphical interface for the user.

Table 3
Simulation parameters.

Parameter	Value	Parameter	Value
Number of cellular BS	25	UE transmit power	2 mW
Number of UEs	75	UAV antenna gain	100
UAV altitude (H)	50 m	Number of features	5
UAV max transmit power	200 mW	Number of epochs	1e4
UAV min transmit power	50 mW	ϵ	0.1
Dense urban (η_{LoS}, η_{NLoS})	$(1.6, 23)$ [dB]	Batch size	24
Dense urban (c_1, c_2)	$(12.076, 0.114)$	Replay buffer size	$1e4$
N_0	-90 dBm	γ	0.99
Carrier frequency	2 GHz	α	0.001

In Fig. 6, a circle notation is considered to show the UAV's coverage area as a function of its transmission power to demonstrate the interference level. Black arrows demonstrate the UAV's directions to show the chosen path. The user can turn off the Graphical User Interface (GUI) to speed up the simulation. All models and trajectories can be saved on a drive for future evaluations. Fig. 6 shows a sample snapshot of the simulator with the chosen path. In this sample, the UAV flies from the source cell (BS_0) towards the destination cell (BS_{24}) with varying transmission power. The source code of this simulator is available on the GitHub repository [91]. A sample video of the designed simulator implementing the different approaches in this study is available on YouTube [92]. In the video, we showed the simulation based on the different approaches during the training phase and also after training for the evaluation.

6.2. Convergence of the IRL

To investigate the convergence for Inverse Reinforcement Learning, the hyperdistance between the expert's feature expectation and the learner's feature expectation is considered a metric to show the convergence behavior of the IRL algorithm. Fig. 7 demonstrates this hyperdistance for different iterations. The learning process stops at the iteration where the hyper-distance is getting lower than a predefined threshold (ϵ_{IRL}). The user can choose the desired threshold based on the requirements. If the user chooses $\epsilon_{\text{IRL}} = 0$, the user wants to achieve the same behavior as the expert. Choosing $\epsilon_{\text{IRL}} = 0$ may need infinite iterations for the IRL algorithm to find the optimal weight.

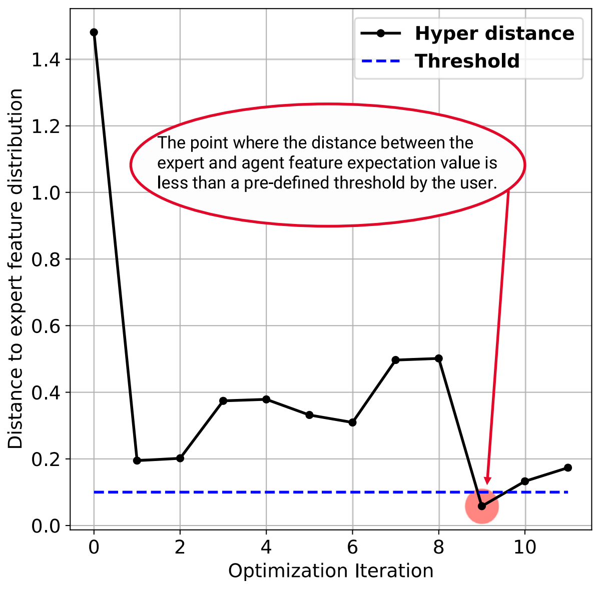


Fig. 7. Hyper distance between the expert and agent feature expectation based on different weights, reward functions, and optimal policy.

In Fig. 7, we assumed that the threshold (ϵ_{IRL}) for the distance between the expert and agent feature expectation is 0.1 and any distance less than the threshold stops the algorithm. After ten iterations, this algorithm stops when the distance is 0.057.

6.3. Convergence of the Q-learning and deep Q-network

Fig. 8 demonstrates a few iterations of the feature policy learning for each unique reward function for the Q-learning with linear function approximation using the SGD. Based on the exploration-greedy rate, the accumulative reward function is converged to the best optimum value. It is worth mentioning that the optimum value for the reward function depends on the obtained weight for the reward function at each iteration based on the QP. Because of that, each plot converges to a different value. Also, Fig. 9 illustrates the same convergence concept for the various iterations of the policy learning for the Deep Q-Network.

6.4. Apprenticeship learning via inverse RL performance - Training phase

All results in this section average over 25 runs for a smooth and clear demonstration for comparison. Figs. 10(a), 10(b), 10(c) represent results for UAV's probabilistic channels, and Figs. 10(d), 10(e), 10(f) show the results for UAV's LoS channels. Figs. 10(a) and 10(d) show the throughput result of the inverse RL using the Q-learning and deep Q-network during the training phase of the optimal reward function after the final optimization and termination of the IRL algorithm. In Fig. 10(d), both approaches converge at the same number of epochs and values after the ϵ-greedy exploration. It is worth mentioning that achieving the maximum throughput is not always the optimum policy since it also increases interference.

Also, Figs. 10(b) and 10(e) show the summation of interference levels on the neighbor UEs when the UAV uses the same resource block for its transmission and the UAV's uplink can interfere with the other UEs' downlink as well. Keeping the interference as low as possible is desired; however, it decreases the throughput as well, and it is not in line with the problem objectives defined in (10a) and (10d) in Section 3. As Fig. 10(e) represents, Q-learning and DQN algorithms converge to different values, and the reason is that because these two algorithms utilized different paths for the problem, thus the agent (UAV) senses a different number of UEs, and the effect of its UL transmission is different in these models.

Figs. 10(c) and 10(f) demonstrate the distance between the location where the UAV finishes its task and the destination cell where it was supposed to finish and stop based on resource limitations such as the battery capacity and flight time. In Fig. 10(c), DQN converges to the distance of zero, which means this algorithm finds the destination cell (in this scenario (BS_{24})). However, the limited energy of UAVs for flight, coupled with the highly dynamic wireless environment and the low convergence power of the Q-learning algorithm, results in an inability to reach the destination in probabilistic channel conditions. However, in Fig. 10(f) both Q-learning and DQN converges to the distance of zero, which means both algorithms find the destination cell.

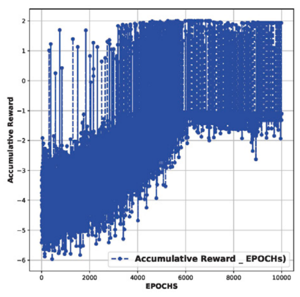
6.5. Apprenticeship learning via inverse RL performance

Here, the goal is to compare the performance of different path planning mechanisms including apprenticeship learning via IRL, inverse deep RL, BC, shortest path, and random action. The previous section shows the result for the inverse RL using Q-learning and DQN during the training phase. Here, based on the trained model and obtained weights for the reward function, a sample scenario is considered to compare the performance of these different approaches. The implemented BC model defined in Section 5.5 predicts the desired action based on the visited features vector state with 89.08% accuracy using the decision tree classification approach.

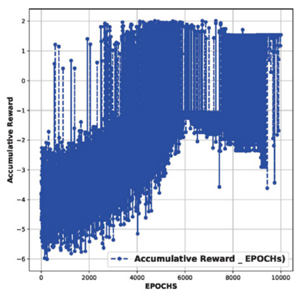
In Fig. 11(a), the throughput of the DQN, BC, shortest path, and random policy is compared. Following BC, which mimics the expert's behavior without knowledge, DQN exhibits better throughput behavior than random and shortest-path algorithms. This can be attributed to the positive influence of feature expectations for the throughput metric on the reward function in the feature state vector ($\phi(s,t)$). In the shortest path, the UAV selects the shortest route between the source cell (BS_0) and the destination to conserve energy, randomly selecting transmission power. Both the movement action and transmission power are random in the random policy. The random policy demonstrates the worst performance in terms of throughput in the same scenario, with the primary factor affecting throughput metrics being the agent's (UAV) transmission power for its UL transmission.

Fig. 11(b) compares interference management across all approaches. This interference represents the summation of interference applied to all neighboring UEs when utilizing the same resource block. This figure's interference level depends on the UAV's transmission power for its UL, the density of neighbor UEs in the neighboring cells, and the distance between the UAV and the affected UEs. BC exhibits the lowest interference level as it precisely mimics the expert's behavior without comprehending it. Following that, the DQN approach has the next level of applied interference. The shortest path selects a route with a higher density of UEs than other approaches, resulting in higher interference than the other two techniques and almost similar to the random policy. If the user aims to train the DQN approach to align more closely with the expert's behavior, a lower value for the epsilon threshold (ϵ_{IRL}) can be chosen. In this case, the algorithm's convergence process will take longer to meet the threshold. However, the agent's feature expectation vector ($\mu(\pi_i)$) will be closer to the expert's feature expectation vector ($\bar{\mu}(\pi_E)$). Therefore, the threshold value can be determined based on the user's expectations for the learning and optimization process of the reward function's weights and the optimal policy. It is essential to note that changes in the epsilon threshold (ϵ_{IRL}) may impact the results in Fig. 7, and more iterations may be required to meet the criteria.

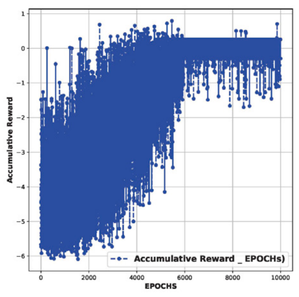
Fig. 12 illustrates the distance between the destination cell (BS_{24}) and the cell where the agent stops or completes its tasks. In both behavioral cloning and the shortest path approaches, the destination cell is reached at the 6th step. However, in the DQN approach, the destination cell is visited at the 7th step. Notably, the random policy does not reach the destination cell due to limitations in battery life and flight time.



(a) Accumulative reward in each episode at iteration 1

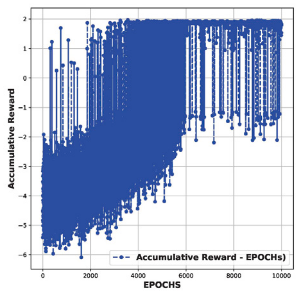


(b) Accumulative reward in each episode at iteration 2

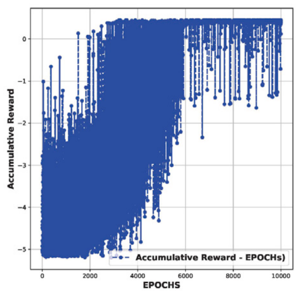


(c) Accumulative reward in each episode at iteration 3

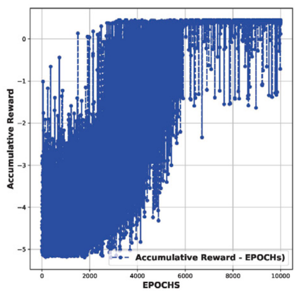
Fig. 8. Convergence of the policy learning using Q-Learning in different iterations for the IRL approach.



(a) Accumulative reward in each episode at iteration 1

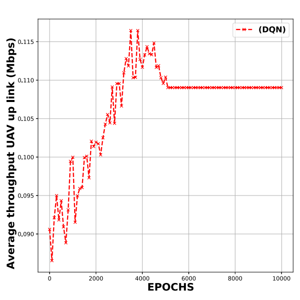


(b) Accumulative reward in each episode at iteration 2

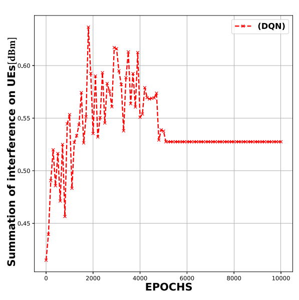


(c) Accumulative reward in each episode at iteration 3

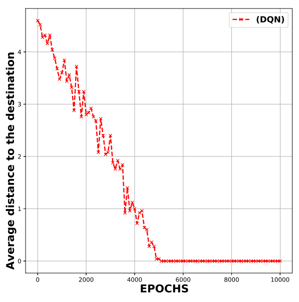
Fig. 9. Convergence of the policy learning using deep Q-Network in different iterations for the IRL approach.



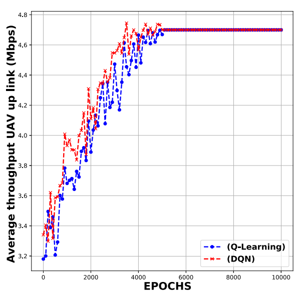
(a) Transmission throughput rate for the UAV's Up-Link in probabilistic channel scenario



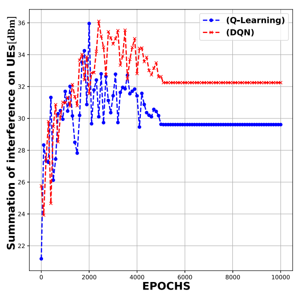
(b) UAV's UL interference on neighbor UEs using the same RBs in probabilistic channel scenario



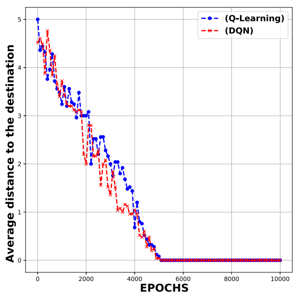
(c) The distance between the last location where the UAV stops its task in probabilistic channel scenario



(d) Transmission throughput rate for the UAV's Up-Link in LoS channel scenario

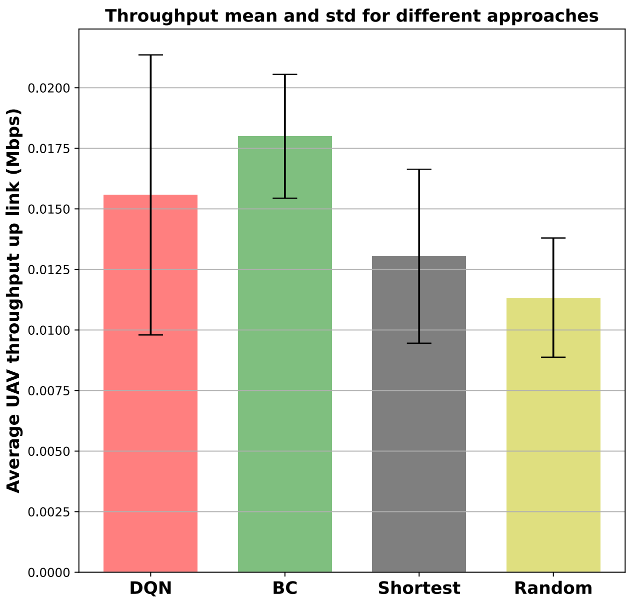


(e) UAV's UL interference on neighbor UEs using the same RBs in LoS channel scenario

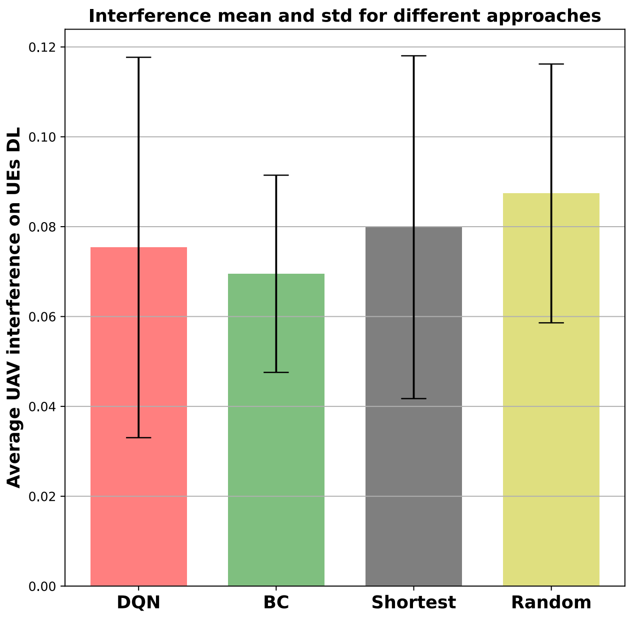


(f) The distance between the last location where the UAV stops its task in LoS channel scenario

Fig. 10. Evaluation of UAV's UL throughput, interference value on neighbor UEs, and the final distance of the UAV during the training phase of the inverse RL for both Q-learning and DQN.



(a) Throughput mean and standard deviation comparison for the UAV Up-link



(b) Average and standard deviation interference affecting the ground neighbor UEs

Fig. 11. Evaluation of the UAV's UL throughput and the effect of interference with neighbor UEs in a single scenario.

6.6. *Inverse RL and behavioral cloning in unseen states by the expert*

In this section, we explore another scenario to assess the performance of the Q-learning, DQN, and BC in a situation where the UAV is placed in a cell that an expert has never seen or experienced. Since we assumed that the expert is only available for a few trajectories or it is costly to have on-demand access to the expert, it is not possible to ask the expert to experience the new state. Therefore, methods like DAGGER are not practical in this situation. To simulate this scenario, an environmental variable, such as wind, is applied to the drone and placed the drone in the adjacent cell (BS_5) as the initial or source cell. The expert has never observed BS_5, and no data exists for this location.

Fig. 13 illustrates trajectories, states, and actions taken by the agent using the three approaches. In Fig. 13(a), the UAV employing the BC approach initiates its journey from BS5 and selects a path with a higher UE density. However, it fails to reach the destination cell (BS24) to complete its task. In Fig. 13(b), Q-learning is utilized to avoid areas with high UE density. Nevertheless, it struggles to reach the destination due to a complex and dynamic wireless environment. On the other hand, DQN successfully identifies the destination cell, navigating around areas with high UE density, as depicted in Fig. 13(c).

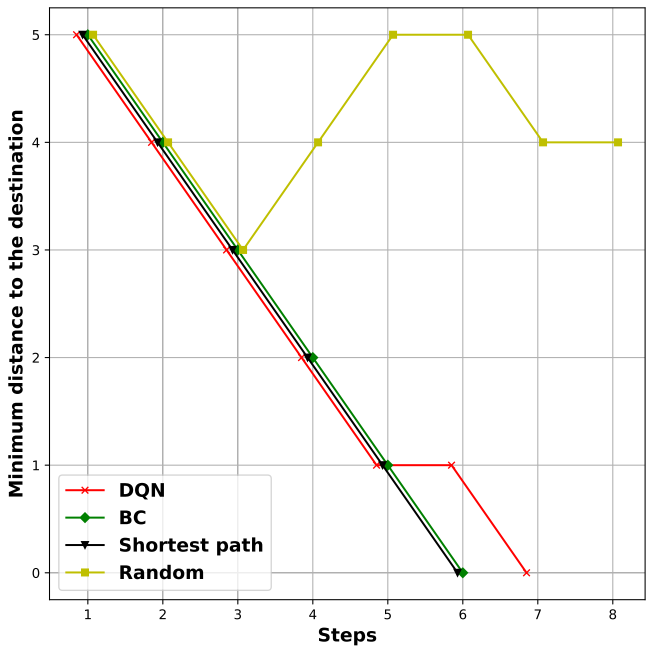
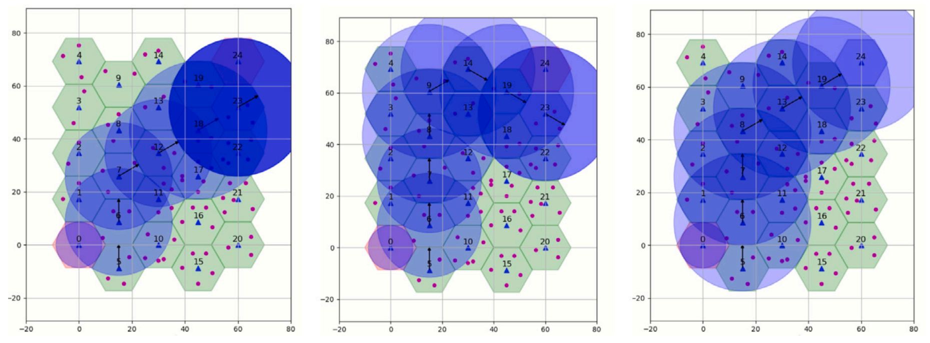


Fig. 12. The distance between the last location where the UAV stops its task and the center of the destination cell.

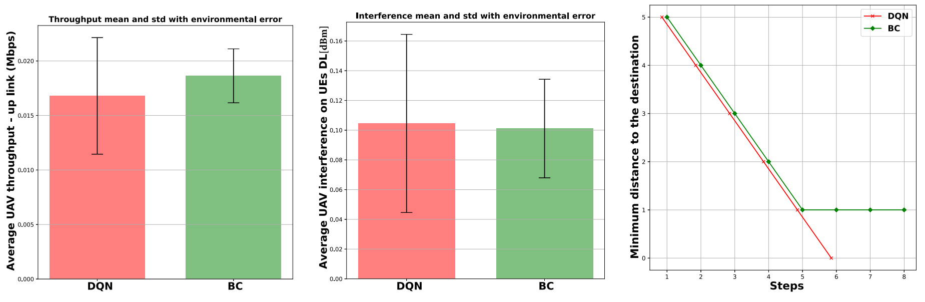
The performance evaluation of throughput, interference, and the distance between the drone and the destination cell is illustrated in Fig. 14. The key focus is completing the agent's task while maintaining an acceptable level of interference and throughput. The DQN approach exhibits a superior average and standard deviation for throughput, as seen in Fig. 14(a). This improvement is attributed to the fact that BC selects power values for its path, which is incorrect. The path has never been experienced before, and data for it was unavailable in the dataset. Fig. 14(b) demonstrates that DQN excels in interference management in the event of errors in the system. Additionally, Fig. 14(c) shows that DQN successfully reached the destination and completed the task by the 6th step. However, BC could not finish the task by reaching the destination cell. In Fig. 14, we demonstrate that the BC method, which accurately mimics the expert's behavior but lacks a comprehensive understanding, achieves slightly higher throughput and lower interference. However, it fails to complete the task and reach the destination. This underscores the superiority of the DQN method, which accomplishes the task and exhibits significantly better performance. Furthermore, this figure also illustrates results in unseen scenarios where BC fails to reach the destination and complete the task, highlighting its limitations in handling such states. This evaluation shows that apprenticeship learning using inverse deep RL performs better compared to imitation learning like BC for the cases where the expert did not experience the visited state by the agent.

One future direction for this work is to explore a more realistic scenario by modeling the antenna gains. Ground-based BS antennas are typically down-tilted and optimized to enhance capacity for ground UEs while minimizing inter-cell interference. Consequently, cellular-connected UAVs may be served by the sidelobes of terrestrial BSs. In such situations, the UAV may establish connections with BSs located at a distance, leading to non-trivial behavior in channel state information.



(a) Path of the behavioral cloning with the applied error (b) Path of the Q-learning with the applied error (c) Path of the deep Q-network with the applied error

Fig. 13. Running Q-learning, DQN, and behavioral cloning on the defined scenario with an environmental error where wind moves the UAV to from cell(0) to the adjacent cell.



(a) Throughput mean and standard deviation for the UAV UL (b) Interference mean and standard deviation on UEs DLs (c) Distance between the UAV and the destination point

Fig. 14. Performance of DQN, and BC with the applied error.

Previous studies, like [93,94], have delved into antenna behavior for trajectory optimization and observation, employing both simulation and experimental approaches. Addressing this challenge could shape future directions regarding system modeling and methodology within this paper.

7. Conclusions

Given the growing applications of unmanned aerial vehicles, the demand for reliable communication technologies that enable low-latency, high-transmission-rate communication is paramount. Cellular networks, particularly 5G and beyond, offer numerous advantages for drones, including reliability, extensive coverage, and security. However, the increasing use of drones as aerial cellular users presents unique challenges, such as potential interference with terrestrial users and base stations. This paper introduces a novel interference-aware scheme that combines joint path planning and power allocation for autonomous cellular-connected unmanned aerial vehicles (UAVs). Our UAVs are tasked with navigating from their initial points to destinations while minimizing interference with terrestrial User Equipment (UEs) and maximizing uplink throughput. To tackle this challenge, we propose an apprenticeship learning approach using IRL, which incorporates both Q-learning and deep Q-network solutions. Additionally, we employ behavioral cloning, a form of imitation learning employing decision tree supervised learning, for comparative analysis.

Our numerical and simulation results demonstrate that apprenticeship learning via IRL closely aligns with expert behavior and outperforms the behavioral cloning approach. Moreover, we allow users to define a threshold to control the agent's adherence to the desired expert behavior. Importantly, our study underscores the effectiveness of inverse RL in scenarios involving system errors or situations where the expert has not collected supervised data for visited states. Furthermore, our research addresses a critical need in the field of UAV communication, offering a solution that optimizes communication quality while mitigating interference. By combining state-of-the-art techniques in reinforcement learning, we contribute valuable insights and methodologies to enhance the reliability and performance of cellular-connected UAVs. This work paves the way for improved drone applications and further advancements in autonomous aerial communication systems.

CRediT authorship contribution statement

Alireza Shamsoshoara: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Formal analysis, Data curation, Conceptualization. **Fatemeh Lotfi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Formal analysis, Data curation, Conceptualization. **Sajad Mousavi:** Writing – original draft, Methodology. **Fatemeh Afghah:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization. **İsmail Güvenç:** Supervision, Resources, Methodology, Investigation, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Alireza Shamsoshoara reports financial support was provided by Air Force Office of Scientific Research. Fatemeh Afghah reports financial support was provided by Air Force Office of Scientific Research. Fatemeh Loti reports financial support was provided by National Science Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research, United States under award number FA9550-20-1-0090 and the National Science Foundation, United States under Grant Numbers CNS-2232048, CNS-2318726, CNS-2332834 and CNS-2120485.

Appendix. Time complexity analysis

Detailed analysis of the time complexity for each algorithm used in the proposed solution is provided here.

A.1. Algorithm 1: Apprenticeship learning via IRL algorithm

To analyze the time complexity of Algorithm 1, we examine each step and its computational complexity.

- **Initialization**: Loading the simulation environment and expert trajectories: Assume this has a constant time complexity, $O(1)$. Calculating the expert feature expectation: This involves summing over N trajectories and the steps t in each trajectory. Assuming each trajectory has an average length L, the complexity is $O(N \times L)$.
- **Main Loop (While True)**: Solving the Quadratic Program (QP) to find the weights has a complexity of $O(QP)$, which depends on the specific QP solver used. Normalizing the weights has a complexity of $O(d)$, where d is the dimension of the weight vector. Training the policy with the learner involves training the policy, which could be a complex operation depending on the specific learner used (e.g., training a neural network). Let us denote this as $O(Learner)$. Resetting the simulation environment is $O(1)$. Running the simulation to calculate the agent's feature expectation has a complexity of $O(M \times L)$, where M is the number of episodes run in the simulation. Calculating the hyper distance involves a dot product calculation, which has a complexity of $O(d)$. Checking the stopping condition is $O(1)$.

The main loop runs until the stopping condition $D < \epsilon_{\text{IRL}}$ is met. Let us denote the number of iterations of the loop as I. Inside each iteration, the most significant contributions to the complexity are solving the QP, training the learner, and running the simulation. Thus, the overall time complexity of the algorithm can be approximated as:

$$O(T \times L) + I \times (O(QP) + O(d) + O(Learner) + O(M \times L) + O(d)). \quad \text{(A.1)}$$

Simplifying this expression, we get:

$$O(T \times L) + I \times (O(QP) + O(Learner) + O(M \times L) + O(d)). \quad \text{(A.2)}$$

A.2. Algorithm 2: Q-learning algorithm with Linear Function Approximation

Similarly, to analyze the time complexity of Algorithm 2, we examine each step and its computational complexity.

- **Initialization**: Loading the simulation environment: This has a constant time complexity, $O(1)$.
- **Main Loop (While episode** $<$ NUM_EPS**)**: The outer loop runs for a specified number of episodes, denoted a NUM_EPS. Within each episode, the distance is reset, and the UAV is reset, both having constant time complexity, $O(1)$.
- **Inner Loop (While distance** $<$ **DIST_LIM & not DONE)**: This loop runs until the distance reaches DIST_LIM or the task is marked as done. Let us assume the maximum number of steps per episode is N.
- **Action Selection**: Selecting an action (either random or greedy) has a constant time complexity, $O(1)$.
- **Environment Update and Reward Calculation**: Updating the UAV's location and calculating SNR, interference, and throughput are assumed to have constant time complexity, $O(1)$. Calculating the features $\boldsymbol{\phi}(s, a, s')$ and the immediate reward $R(s) = \boldsymbol{w}^T \cdot \boldsymbol{\phi}(s')$ both have constant time complexity, $O(d)$, where d is the number of features.
- **Q-value Calculation and Update**: Predicting $Q(s')$ using the SGD model has a complexity of $O(d)$. Updating the Q-value $Q^+(s, a)$ involves a maximum operation and an addition, both of which have a constant time complexity, $O(1)$. Updating the SGD model has a complexity of $O(d)$.
- **Epsilon Decay**: Decaying epsilon has a constant time complexity, $O(1)$.

Finally, the overall time complexity of the algorithm can be approximated as follows:

- **Initialization**: $O(1)$.
- **Outer Loop (Episodes)**: This runs for NUM_EPS iterations.
- **Inner Loop (Steps per Episode)**: This runs for up to N steps per episode.

Thus, the time complexity within the inner loop for each step includes:

- Action selection: $O(1)$.
- Environment update and reward calculation: $O(d)$.
- Q-value prediction and update: $O(d)$.
- Epsilon decay (outside inner loop but within the outer loop): $O(1)$.

Combining these, the time complexity per episode is:

$$O(N \times (O(1) + O(d) + O(d) + O(1))) = O(N \times d). \quad \text{(A.3)}$$

Therefore, the overall time complexity for NUM_EPS episodes is:

$$O(\text{NUM\_EPS} \times N \times d). \quad \text{(A.4)}$$

A.3. Algorithm 3: Deep Q-network to predict the Q-action-value on batch samples

To analyze the time complexity of Algorithm 3, we examine each step and its computational complexity.

- **Initialization**: Same as Algorithm 2, initialization of variables and the model is $O(1)$.
- **Main Loop (While episode** $<$ **NUM_EPS)**: The outer loop runs for a specified number of episodes, denoted as NUM_EPS.
- **Inner Loop (While distance** $<$ **DIST_LIM & not DONE)**: This loop runs until the distance reaches DIST_LIM or the task is marked as done. We assume the maximum number of steps per episode is N.
- **Collecting Features, Actions, Reward for Replay Memory**: Collecting features, actions, and rewards has a complexity of $O(1)$ for each step.
- **Batch Sampling and Training**: Once the episode exceeds NUM_EPS/10, a batch of size $BATCH\_SIZE$ is randomly sampled from the replay memory. The cost of sampling a batch is $O(BATCH\_SIZE)$.
- **Processing Each Batch**: For each data point in the batch, extracting x_train has a complexity of $O(1)$. Calculating $Q^+(s, a)$ involves either setting it to $R(s)$ or calculating $R(s) + \gamma \max_{a'} Q(s', a')$, both of which have a complexity of $O(1)$. Storing y_train has a complexity of $O(1)$.

- **Model Training**: The model training step involves fitting the DQN model to the x_train and y_train batches. Let f be the time complexity of fitting the model for each batch. This step's complexity is $O(f \times BATCH\_SIZE)$.
- **Epsilon Decay**: Decaying epsilon has a constant time complexity, $O(1)$.

As a result, the overall time complexity of the algorithm can be approximated as follows:

- **Initialization**: $O(1)$.
- **Outer Loop (Episodes)**: This runs for NUM_EPS iterations.
- **Inner Loop (Steps per Episode)**: This runs for up to N steps per episode.

Within the inner loop for each step, after NUM_EPS/10:

- Collecting features, actions, and rewards: $O(1)$.
- Sampling a batch: $O(BATCH\_SIZE)$.
- Processing each batch:
  - Extracting x_train: $O(1)$.
  - Calculating $Q^{+}(s,a)$: $O(1)$.
  - Storing y_train: $O(1)$.
- Model training: $O(f \times BATCH\_SIZE)$.

Combining these, the time complexity per step after NUM_EPS/10 is:

$$\begin{aligned}&O(BATCH\_SIZE) + O(BATCH\_SIZE \times (O(1)+O(1)+O(1)))\\&+ O(f \times BATCH\_SIZE) = O(BATCH\_SIZE)\\&+ O(f \times BATCH\_SIZE)\\&= O(BATCH\_SIZE \times (1+f)).\end{aligned} \tag{A.5}$$

Since this occurs for each step in the inner loop, the time complexity per episode is:

$$\begin{aligned}&O(N \times (1 + BATCH\_SIZE \times (1+f)))\\&\quad = O(N \times BATCH\_SIZE \times (1+f)).\end{aligned} \tag{A.6}$$

Therefore, the overall time complexity for NUM_EPS episodes is:

$$O(\text{NUM\_EPS} \times N \times BATCH\_SIZE \times (1+f)). \tag{A.7}$$

References

[1] A. Andreeva-Mori, D. Kubo, K. Kobayashi, Y. Okuno, J.R. Homola, M. Johnson, P.H. Kopardekar, Supporting disaster relief operations through UTM: Operational concept and flight tests of unmanned and manned vehicles at a disaster drill, in: AIAA Scitech 2020 Forum, 2020, p. 2202.

[2] A. Pandey, P.K. Shukla, R. Agrawal, An adaptive Flying Ad-hoc Network (FANET) for disaster response operations to improve quality of service (QoS), Modern Phys. Lett. B 34 (10) (2020) 2050010.

[3] S. Javed, A. Hassan, R. Ahmad, W. Ahmed, M.M. Alam, J.J. Rodrigues, UAV trajectory planning for disaster scenarios, Veh. Commun. (2023) 100568.

[4] F. Afghah, A. Razi, J. Chakareski, J. Ashdown, Wildfire monitoring in remote areas using autonomous unmanned aerial vehicles, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, 2019, pp. 835–840, http://dx.doi.org/10.1109/INFCOMW.2019.8845309.

[5] A. Shamsoshoara, F. Afghah, A. Razi, L. Zheng, P.Z. Fulé, E. Blasch, Aerial Imagery Pile burn detection using deep learning: the FLAME dataset, Comput. Netw. (2021) 108001.

[6] M. Keshavarz, A. Shamsoshoara, F. Afghah, J. Ashdown, A real-time framework for trust monitoring in a network of unmanned aerial vehicles, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, IEEE, 2020, pp. 677–682.

[7] H. Rajoli, P. Afshin, F. Afghah, Thermal image calibration and correction using unpaired cycle-consistent adversarial networks, in: IEEE 57th Asilomar Conference on Signals, Systems, and Computers, 2023, pp. 1425–1429.

[8] H. Rajoli, S. Khoshdel, F. Afghah, X. Ma, FlameFinder: Illuminating obscured fire through smoke with attentive deep metric learning, IEEE Transactions on Geoscience and Remote Sensing (2024) http://dx.doi.org/10.1109/TGRS.2024.3440880, 1-1.

[9] M. Mozaffari, W. Saad, M. Bennis, M. Debbah, Unmanned aerial vehicle with underlaid device-to-device communications: Performance and tradeoffs, IEEE Trans. Wireless Commun. 15 (6) (2016) 3949–3963.

[10] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, M. Debbah, A tutorial on UAVs for wireless networks: Applications, challenges, and open problems, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2334–2360.

[11] Q. Huang, A. Razi, F. Afghah, P. Fule, Wildfire spread modeling with aerial image processing, in: 2020 IEEE 21st International Symposium on "a World of Wireless, Mobile and Multimedia Networks", WoWMoM, 2020, pp. 335–340, http://dx.doi.org/10.1109/WoWMoM49955.2020.00063.

[12] Y. Zeng, R. Zhang, T.J. Lim, Wireless communications with unmanned aerial vehicles: Opportunities and challenges, IEEE Commun. Mag. 54 (5) (2016) 36–42.

[13] N. Namvar, F. Afghah, Heterogeneous airborne mmWave cells: Optimal placement for power-efficient maximum coverage, in: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, 2022, pp. 1–6, http://dx.doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798023.

[14] N. Namvar, F. Afghah, Joint 3D placement and interference management for drone small cells, in: 2021 55th Asilomar Conference on Signals, Systems, and Computers, 2021, pp. 780–784, http://dx.doi.org/10.1109/IEEECONF53345.2021.9723350.

[15] 3rd Generation Partnership Project (3GPP), 3Rd generation partnership project (3GPP), release 15, 2020, https://www.3gpp.org/specifications/67-releases. (Accessed 05 December 2020).

[16] 3rd Generation Partnership Project (3GPP), 3GPP TS 36.420 3GPP TSG RAN evolved universal terrestrial radio access network (EUTRAN), X2 general aspects and principles, version 11.0.0, release 11, 2012, https://www.3gpp.org/specifications/67-releases. (Accessed on 15/17/2020).

[17] A. Rovira-Sugranes, A. Razi, F. Afghah, J. Chakareski, A review of AI-enabled routing protocols for UAV networks: Trends, challenges, and future outlook, Ad Hoc Netw. 130 (2022) 102790.

[18] M.-A. Lahmeri, M.A. Kishk, M.-S. Alouini, Artificial intelligence for UAV-enabled wireless networks: A survey, IEEE Open J. Commun. Soc. 2 (2021) 1015–1040.

[19] Y. Huang, Q. Wu, R. Lu, X. Peng, R. Zhang, Massive MIMO for cellular-connected UAV: Challenges and promising solutions, IEEE Commun. Mag. 59 (2) (2021) 84–90.

[20] M. Gharib, B. Hopkins, J. Murrin, A. Koka, F. Afghah, 5G wings: Investigating 5G-connected drones performance in non-urban areas, in: 2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, IEEE, 2023, pp. 1–6.

[21] A. Festag, S. Udupa, L. Garcia, R. Wellens, M. Hecht, P. Ulfig, End-to-end performance measurements of drone communications in 5g cellular networks, in: 2021 IEEE 94th Vehicular Technology Conference, VTC2021-Fall, IEEE, 2021, pp. 1–6.

[22] A.H.F. Raouf, S.J. Maeng, I. Guvenc, Ö. Özdemir, M. Sichitiu, Cellular spectrum occupancy probability in urban and rural scenarios at various UAS altitudes, in: 2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, IEEE, 2023, pp. 1–6.

[23] S.J. Maeng, O. Ozdemir, I. Guvenc, M.L. Sichitiu, M. Mushi, R. Dutta, LTE I/Q data set for UAV propagation modeling, communication, and navigation research, IEEE Commun. Mag. 61 (9) (2023) 90–96.

[24] Y. Huo, X. Dong, W. Xu, 5G cellular user equipment: From theory to practical hardware design, IEEE Access 5 (2017) 13992–14010.

[25] A. Checko, H.L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M.S. Berger, L. Dittmann, Cloud RAN for mobile networks—A technology overview, IEEE Commun. Surv. Tutor. 17 (1) (2014) 405–426.

[26] M. Gharib, S. Nandadapu, F. Afghah, An exhaustive study of using commercial LTE network for UAV communication in rural areas, 2021, arXiv:2105.03778.

[27] F. Lotfi, O. Semiari, Performance analysis and optimization of uplink cellular networks with flexible frame structure, in: 2021 IEEE 93rd Vehicular Technology Conference, VTC2021-Spring, 2021, pp. 1–5, http://dx.doi.org/10.1109/VTC2021-Spring51267.2021.9448665.

[28] 3rd Generation Partnership Project (3GPP), 5G;Unmanned Aerial System (UAS) support in 3gpp, 3GPP TS 22.125 version 16.3.0 release 16, technical specification, 2020, https://www.3gpp.org/specifications/67-releases.

[29] 3rd Generation Partnership Project (3GPP), Universal [mobile telecommunications system] (UMTS); LTE; 5G; T8 reference point for Northbound APIs, 3GPP TS 29.122 version 17.5.0 release 17, technical specification, 2022, https://www.3gpp.org/specifications/67-releases.

[30] D. Mishra, E. Natalizio, A survey on cellular-connected UAVs: Design challenges, enabling 5G/B5G innovations, and experimental advancements, Comput. Netw. 182 (2020) 107451.

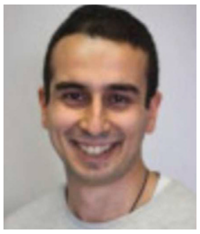
[31] F. Lotfi, O. Semiari, W. Saad, Semantic-aware collaborative deep reinforcement learning over wireless cellular networks, in: ICC 2022 - IEEE International Conference on Communications, 2022, pp. 5256–5261, http://dx.doi.org/10.1109/ICC45855.2022.9839122.

[32] F. Lotfi, O. Semiari, F. Afghah, Evolutionary deep reinforcement learning for dynamic slice management in O-RAN, 2022, arXiv preprint arXiv:2208.14394.

[33] F. Lotfi, F. Afghah, J. Ashdown, Attention-based open RAN slice management using deep reinforcement learning, in: GLOBECOM 2023 - 2023 IEEE Global Communications Conference, 2023, pp. 6328–6333, http://dx.doi.org/10.1109/GLOBECOM54140.2023.10436850.

[34] F. Lotfi, F. Afghah, Open RAN LSTM traffic prediction and slice management using deep reinforcement learning, in: 2023 57th Asilomar Conference on Signals, Systems, and Computers, 2023, pp. 646–650, http://dx.doi.org/10.1109/IEEECONF59524.2023.10476972.
[35] A. Shamsoshoara, M. Khaledi, F. Afghah, A. Razi, J. Ashdown, K. Turck, A solution for dynamic spectrum management in mission-critical UAV networks, in: 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON, IEEE, 2019, pp. 1–6.
[36] A. Shamsoshoara, F. Afghah, A. Razi, S. Mousavi, J. Ashdown, K. Turk, An autonomous spectrum management scheme for unmanned aerial vehicle networks in disaster relief operations, IEEE Access 8 (2020) 58064–58079.
[37] Y. Zeng, X. Xu, S. Jin, R. Zhang, Simultaneous navigation and radio mapping for cellular-connected UAV with deep reinforcement learning, IEEE Trans. Wireless Commun. (2021).
[38] Y. Zeng, X. Xu, Path design for cellular-connected UAV with reinforcement learning, in: 2019 IEEE Global Communications Conference, GLOBECOM, IEEE, 2019, pp. 1–6.
[39] R. Xie, Z. Meng, L. Wang, H. Li, K. Wang, Z. Wu, Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments, IEEE Access 9 (2021) 24884–24900.
[40] C. Chronis, G. Anagnostopoulos, E. Politi, A. Garyfallou, I. Varlamis, G. Dimitrakopoulos, Path planning of autonomous UAVs using reinforcement learning, in: Journal of Physics: Conference Series, Vol. 2526, IOP Publishing, 2023, 012088.
[41] U. Challita, W. Saad, C. Bettstetter, Deep reinforcement learning for interference-aware path planning of cellular-connected UAVs, in: 2018 IEEE International Conference on Communications, ICC, IEEE, 2018, pp. 1–7.
[42] U. Challita, W. Saad, C. Bettstetter, Interference management for cellular-connected UAVs: A deep reinforcement learning approach, IEEE Trans. Wireless Commun. 18 (4) (2019) 2125–2140.
[43] J. Bao, Y. Yang, Y. Wang, X. Yang, Z. Du, Path planning for cellular-connected UAV using heuristic algorithm and reinforcement learning, in: 2023 25th International Conference on Advanced Communication Technology, ICACT, IEEE, 2023, pp. 454–459.
[44] X. Zhu, L. Wang, Y. Li, S. Song, S. Ma, F. Yang, L. Zhai, Path planning of multi-UAVs based on deep Q-network for energy-efficient data collection in UAVs-assisted IoT, Veh. Commun. 36 (2022) 100491.
[45] Y. Li, P. Ni, V. Chang, Application of deep reinforcement learning in stock trading strategies and stock forecasting, Computing 102 (6) (2020) 1305–1322.
[46] B. Huang, Z. Li, Y. Xu, L. Pan, S. Wang, H. Hu, V. Chang, Deep reinforcement learning for performance-aware adaptive resource allocation in mobile edge computing, Wirel. Commun. Mob. Comput. 2020 (2020) 1–17.
[47] G.F. Riley, T.R. Henderson, The ns-3 network simulator, in: Modeling and Tools for Network Simulation, Springer, 2010, pp. 15–34.
[48] I. Gomez-Miguelez, A. Garcia-Saavedra, P.D. Sutton, P. Serrano, C. Cano, D.J. Leith, srsLTE: An open-source platform for LTE evolution and experimentation, in: Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization, 2016, pp. 25–32.
[49] N. Nikaein, M.K. Marina, S. Manickam, A. Dawson, R. Knopp, C. Bonnet, OpenAirInterface: A flexible platform for 5G research, ACM SIGCOMM Comput. Commun. Rev. 44 (5) (2014) 33–38.
[50] A. Shamsoshoara, F. Afghah, E. Blasch, J. Ashdown, M. Bennis, UAV-assisted communication in Remote Disaster Areas using imitation learning, IEEE Open J. Commun. Soc. (2021).
[51] P. Abbeel, A.Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proceedings of the Twenty-First International Conference on Machine Learning, 2004, p. 1.
[52] S. Arora, P. Doshi, A survey of inverse reinforcement learning: Challenges, methods and progress, Artificial Intelligence 297 (2021) 103500.
[53] A. Shamsoshoara, M. Khaledi, F. Afghah, A. Razi, J. Ashdown, Distributed cooperative spectrum sharing in uav networks using multi-agent reinforcement learning, in: 2019 16th IEEE Annual Consumer Communications & Networking Conference, CCNC, IEEE, 2019, pp. 1–6.
[54] H. Kurunathan, H. Huang, K. Li, W. Ni, E. Hossain, Machine learning-aided operations and communications of unmanned aerial vehicles: A contemporary survey, IEEE Commun. Surv. Tutor. (2023).
[55] S. Aggarwal, N. Kumar, Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges, Comput. Commun. 149 (2020) 270–299.
[56] W. Mei, Q. Wu, R. Zhang, Cellular-connected UAV: Uplink association, power control and interference coordination, IEEE Trans. Wireless Commun. 18 (11) (2019) 5380–5393.
[57] S. Hu, X. Yuan, W. Ni, X. Wang, Trajectory planning of cellular-connected UAV for communication-assisted radar sensing, IEEE Trans. Commun. 70 (9) (2022) 6385–6396.
[58] D. Yang, Q. Dan, L. Xiao, C. Liu, L. Cuthbert, An efficient trajectory planning for cellular-connected UAV under the connectivity constraint, China Commun. 18 (2) (2021) 136–151.
[59] S. Zhang, Y. Zeng, R. Zhang, Cellular-enabled UAV communication: Trajectory optimization under connectivity constraint, in: 2018 IEEE International Conference on Communications, ICC, IEEE, 2018, pp. 1–6.
[60] E. Bulut, I. Guevenc, Trajectory optimization for cellular-connected UAVs with disconnectivity constraint, in: 2018 IEEE International Conference on Communications Workshops, ICC Workshops, IEEE, 2018, pp. 1–6.
[61] S. Li, G. Liu, K. Zhang, Z. Qian, S. Ding, DRL-based joint path planning and jamming power allocation optimization for suppressing netted radar system, IEEE Signal Process. Lett. 30 (2023) 548–552, http://dx.doi.org/10.1109/LSP.2023.3270762.
[62] G. Zhao, Y. Wang, T. Mu, Z. Meng, Z. Wang, Reinforcement learning assisted multi-UAV task allocation and path planning for IIoT, IEEE Internet Things J. (2024) 1.
[63] X. Zheng, Y. Wu, L. Zhang, M. Tang, F. Zhu, Priority-aware path planning and user scheduling for UAV-mounted MEC networks: A deep reinforcement learning approach, Phys. Commun. 62 (2024) 102234.
[64] P. Qin, Y. Fu, J. Zhang, S. Geng, J. Liu, X. Zhao, DRL-based resource allocation and trajectory planning for NOMA-enabled multi-UAV collaborative caching 6 G network, IEEE Trans. Veh. Technol. (2024) 1–15.
[65] X. Luo, C. Chen, C. Zeng, C. Li, J. Xu, S. Gong, Deep reinforcement learning for joint trajectory planning, transmission scheduling, and access control in UAV-assisted wireless sensor networks, Sensors 23 (10) (2023) 4691.
[66] A.M. Jwaifel, T. Van Do, Deep reinforcement learning for jointly resource allocation and trajectory planning in UAV-assisted networks, in: N.T. Nguyen, J. Botzheim, L. Gulyás, M. Núñez, J. Treur, G. Vossen, A. Kozierkiewicz (Eds.), Computational Collective Intelligence, Springer Nature Switzerland, Cham, 2023, pp. 71–83.
[67] Q. Wu, Y. Zeng, R. Zhang, Joint trajectory and communication design for multi-UAV enabled wireless networks, IEEE Trans. Wireless Commun. 17 (3) (2018) 2109–2121.
[68] A. Al-Hourani, S. Kandeepan, S. Lardner, Optimal LAP altitude for maximum coverage, IEEE Wirel. Commun. Lett. 3 (6) (2014) 569–572.
[69] M. Bain, C. Sammut, A framework for behavioural cloning, in: Machine Intelligence 15, 1995, pp. 103–129.
[70] T. Osa, J. Pajarinen, G. Neumann, J.A. Bagnell, P. Abbeel, J. Peters, et al., An algorithmic perspective on imitation learning, Found. Trends® Robot. 7 (1–2) (2018) 1–179.
[71] S. Adams, T. Cody, P.A. Beling, A survey of inverse reinforcement learning, Artif. Intell. Rev. (2022) 1–40.
[72] S. Ross, G. Gordon, D. Bagnell, A reduction of imitation learning and structured prediction to no-regret online learning, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.
[73] A.Y. Ng, S.J. Russell, et al., Algorithms for inverse reinforcement learning, in: Icml, Vol. 1, 2000, p. 2.
[74] N.D. Ratliff, J.A. Bagnell, M.A. Zinkevich, Maximum margin planning, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 729–736.
[75] M. Wulfmeier, P. Ondruska, I. Posner, Maximum entropy deep inverse reinforcement learning, 2015, arXiv preprint arXiv:1507.04888.
[76] Z. Zhou, M. Bloem, N. Bambos, Infinite time horizon maximum causal entropy inverse reinforcement learning, IEEE Trans. Autom. Control 63 (9) (2017) 2787–2802.
[77] S. Diamond, S. Boyd, CVXPY: A python-embedded modeling language for convex optimization, J. Mach. Learn. Res. 17 (83) (2016) 1–5.
[78] CVXOPT, Convex Optimization for Python, https://cvxopt.org/. (Accessed 04 July 2021).
[79] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[80] S. Levine, A. Kumar, G. Tucker, J. Fu, Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020, arXiv preprint arXiv:2005.01643.
[81] Y. Shoham, R. Powers, T. Grenager, Multi-Agent Reinforcement Learning: A Critical Survey, Tech. Rep., Technical Report, Stanford University, 2003.
[82] L. Busoniu, R. Babuska, B. De Schutter, Multi-agent reinforcement learning: A survey, in: 2006 9th International Conference on Control, Automation, Robotics and Vision, IEEE, 2006, pp. 1–6.
[83] Y. Li, Y. Yuan, Convergence analysis of two-layer neural networks with relu activation, in: Advances in Neural Information Processing Systems, 2017, pp. 597–607.
[84] D.M. Allen, Mean square error of prediction as a criterion for selecting variables, Technometrics 13 (3) (1971) 469–475.
[85] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[86] F. Torabi, G. Warnell, P. Stone, Behavioral cloning from observation, 2018, arXiv preprint arXiv:1805.01954.
[87] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, CRC Press, 1984.
[88] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[89] A. Al-Hourani, S. Kandeepan, A. Jamalipour, Modeling air-to-ground path loss for low altitude platforms in urban environments, in: 2014 IEEE Global Communications Conference, IEEE, 2014, pp. 2898–2904.
[90] H.K. Armeniakos, K. Maliatsos, P.S. Bithas, A.G. Kanatas, A stochastic geometry-based performance analysis of a UAV corridor-assisted IoT network, Front. Commun. Netw. 5 (2024) 1337697.
[91] F. Lotfi, Inverse RL apprenticeship learning UAV communication code, 2024, https://github.com/FLotfiGit/Inverse-RL-Apprenticeship-learning-UAV-Communication.
[92] A. Shamsoshoara, Apprenticeship Learning Via Inverse RL for a Cellular-Connected UAV, Wireless Networking & Information Processing (WINIP) LAB, 2021, https://youtu.be/FGAlHaTQ_nc. (Accessed 19 May 2021).
[93] M.M.U. Chowdhury, S.J. Maeng, E. Bulut, I. Güvenç, 3-D trajectory optimization in UAV-assisted cellular networks considering antenna radiation pattern and backhaul constraint, IEEE Trans. Aerosp. Electron. Syst. 56 (5) (2020) 3735–3750.
[94] M.M.U. Chowdhury, C.K. Anjinappa, I. Guvenc, M. Sichitiu, O. Ozdemir, U. Bhattacherjee, R. Dutta, V. Marojevic, B. Floyd, A taxonomy and survey on experimentation scenarios for aerial advanced wireless testbed platforms, in: 2021 IEEE Aerospace Conference (50100), IEEE, 2021, pp. 1–20.


Alireza Shamsoshoara received the B.Sc. degree in Electrical Engineering from Shahid Beheshti University (SBU), Tehran, Iran, and the M.Sc. degree in Electrical and Communication Engineering from Khaje Nasir Toosi University of Technology (KNTU), Tehran, Iran in 2012 and 2014 respectively. Also, he received the M.Sc. degree in Informatics from Northern Arizona University in 2018. Currently, he is a Ph.D. student in the School of Informatics, Computing & Cyber Systems at Northern Arizona University. His main research interests are security, wireless networks, UAV networks, spectrum sharing, and machine learning.

Fatemeh Lotfi received the B.Sc. degree in Electrical Engineering from Iran University of Science and Technology (IUST), Tehran, Iran, and the M.Sc. degree in Electrical and Communication Engineering from University of Tehran (UT), Tehran, Iran in 2010 and 2013 respectively. Currently, she is a Ph.D. student in the Department of Electrical and Computer Engineering at Clemson University. Her main research interests are wireless networks, UAV networks, semantic communication, deep reinforcement learning and machine learning.

Sajad Mousavi received the B.Sc. degree in computer engineering from Zanjan University (ZNU), Zanjan, Iran, in 2010, and the M.Sc. degree in artificial intelligence and robotics from the Iran University of Science and Technology (IUST), Tehran, Iran, in 2012. He is currently pursuing the Ph.D. degree with the School of Informatics, Computing and Cyber Systems, Northern Arizona University. He worked as a Research Assistant in machine learning and deep learning with the National University of Ireland Galway, Galway, Ireland, from 2015 to 2017. His main research interests include machine learning, deep learning, computer vision, multiagent systems, and task allocation.

Fatemeh Afghah (Senior Member, IEEE) is currently an Associate Professor with the Department of Electrical and Computer Engineering, Clemson University, where she is also the Director of the Intelligent Systems and Wireless Networking (IS-WiN) Laboratory. Prior to joining Clemson University, she was an Associate Professor with the School of Informatics, Computing and Cyber Systems, Northern Arizona University (NAU), Flagstaff, AZ, USA, from 2015 to 2021. She has coauthored more than 100 technical papers. Her research interests include wireless communication networks, decision making in multiagent systems, radio spectrum management, hardware-based security, and artificial intelligence in healthcare. She was a recipient of several awards, including the NSF CRII Award, in 2017, the Air Force Office of Scientific Research Young Investigator Award, in 2019, and the National Science Foundation (NSF) CAREER Award, in 2020. She serves as an Editor for several journals, including Ad hoc Networks journal and Computer Networks journal.

Ismail Guvenc (F'21) received the Ph.D. degree in electrical engineering from the University of South Florida in 2006. He was with the Mitsubishi Electric Research Labs in 2005, with DOCOMO Innovations from 2006 to 2012, and with Florida International University, from 2012 to 2016. From 2016 to 2020, he has been an Associate Professor, and since 2020, he has been a Professor, with the Department of Electrical and Computer Engineering of North Carolina State University. He has published more than 300 conference/journal articles and book chapters, and several standardization contributions. He coauthored/co-edited four books and he is an inventor/coinventor of some 30 U.S. patents. His recent research interests include 5G wireless systems, communications and networking with drones, and heterogeneous wireless networks. Güvenc is a Senior Member of the National Academy of Inventors. He was a recipient of the USF Outstanding Dissertation Award in 2006, the Ralph E. Powe Junior Faculty Enhancement Award in 2014, the NSF CAREER Award in 2015, the FIU College of Engineering Faculty Research Award in 2016, and the NCSU ECE R. Ray Bennett Faculty Fellow Award in 2019. He has served as an Editor for the IEEE Communications Letters from 2010 to 2015 and the IEEE Wireless Communications Letters from 2011 to 2016. He has been serving as an Editor for the IEEE Transactions on Wireless Communications since 2016, and for IEEE Transactions on Communications since 2020. He has served as a guest editor for several other journals.