Learning to Rest: A Q-Learning Approach to Flying Base Station Trajectory Design with Landing Spots

Harald Bayerlein, Rajeev Gangula, and David Gesbert
Communication Systems Department, EURECOM
Sophia Antipolis, France
{harald.bayerlein, rajeev.gangula, gesbert}@eurecom.fr

***Abstract*—We consider the problem of trajectory optimization for an autonomous UAV-mounted base station that provides communication services to ground users with the aid of landing spots (LSs). Recently, the concept of LSs was introduced to alleviate the problem of short mission durations arising from the limited on-board battery budget of the UAV, which severely limits network performance. In this work, using Q-learning, a model-free reinforcement learning (RL) technique, we train a neural network (NN) to make movement decisions for the UAV that maximize the data collected from the ground users while minimizing power consumption by exploiting the landing spots. We show that the system intelligently integrates landing spots into the trajectory to extend flying time and is able to learn the topology of the network over several flying epochs without any explicit information about the environment.**

I. Introduction

The large-scale use of unmanned aerial vehicles (UAVs) is envisioned for a multitude of applications in the society of the future, with "last mile" delivery of goods to consumers being a frequently cited one. Commercial deployment of autonomous drones to deliver food and groceries to customers in Iceland's capital Reykjavik has started very recently, putting the Icelandic air authorities at the top of the list of countries that allow autonomous drone flight [1]. More countries are bound to follow, opening up markets to a variety of applications other than just cellular-connected delivery or survey drones.

One promising idea is to leverage the versatility of drones for the mobile communication infrastructure itself. Deploying mobile base stations (BSs) mounted on UAVs could provide network operators with the capability to react fast and efficiently to sudden demand increases in localized areas, e.g. caused by crowded events, as well as immediately re-establish destroyed networks in disaster and search-and-rescue scenarios. Alternative carrier systems such as fixed-wing aircraft or balloons (as in Google's spin-off Loon) could also be used to establish network coverage and Internet connectivity in the vast areas around the globe currently without it.

No matter the scale of the established network, the Quality of Service (QoS) afforded to the network's users is strongly dependent on the location of the UAV BS. Previous works either addressed the placement problem of finding a drone position that maximizes the system's QoS goals, e.g. in [2], [3], or the trajectory planning problem where the drone's flying path from start to end is optimized with respect to the QoS goals, e.g. in [4]–[7]. When only addressing the placement problem, the performance while flying to and from the designated position is not being optimized. In [5], considering the whole trajectory allows the authors to jointly optimize scheduling and user association, whereas the authors of [4] and [6] consider the power consumption of the UAV and attached BS in addition to the QoS. A large number of works exist that investigate the general challenges and opportunities in wireless communications with UAVs, summaries of which are provided in [8], [9].

As this work focuses on small and versatile multirotor type drones, of which the quadcopter is the most commonly used example, power consumption and battery energy density restrictions are central constraints for UAV BS mission planning. A typical quadcopter fitted with a small base station as used in experiments at Eurecom [10] can only sustain a mission duration of around 15 minutes. As power consumption for flying usually exceeds power consumption of the carried BS by far, the authors in [6] introduced the concept of landing spots (LSs), where a UAV BS can land thus saving energy while continuing to serve users.

Q-learning, a model-free reinforcement learning (RL) technique, has received a lot of attention in a variety of fields since it was employed in a seminal paper by Mnih et al. [11] for stable training of a neural network (NN) playing Atari video games with superhuman performance. The combination of a deep NN and Q-learning was considered to be inherently unstable until then [12]. Reinforcement learning in combination with deep NN training has not been considered widely for challenges in UAV communications. In [13], the authors use RL with a deep recurrent NN to plan paths for a network of cellular-connected UAVs while minimizing interference in the ground network. The concept of optimizing a trajectory of a UAV BS through a NN trained with reinforcement Q-learning was introduced in [7], however without any consideration of power consumption and LSs.

In this work, we consider the UAV acting as a mobile BS serving a group of ground users maximizing the sum of the information rate over the whole flying time with a limited amount of energy in the drone's battery at the start. Movement decisions are therefore made based on the UAV's current position and battery content, as well as the expectation of the

The authors acknowledge the support of the PERFUME project funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 670896).

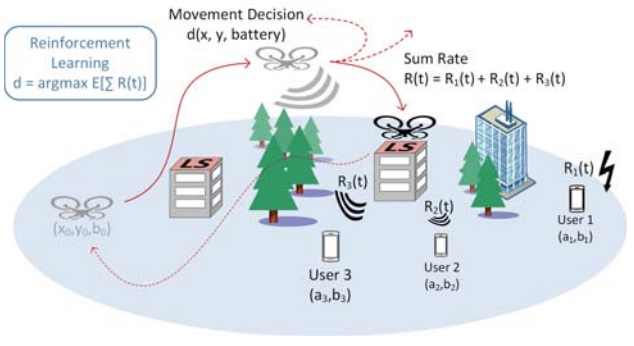

Fig. 1. UAV BS movement decisions are made based on the drone's current state, i.e. position and battery content. While LSs offer the possibility to conserve energy, the UAV BS might have to sacrifice QoS for some users.

total sum rate that can be achieved until the battery has run out. To save energy during the mission, the UAV is allowed to land in designated LSs as depicted in Fig. 1.

II. System Model

We consider a square grid world $\mathcal{G} = [0, g] \times [0, g]$ with the UAV serving K ground users located at positions $(a_k, b_k) \in \mathcal{G}$ with $k \in \{1, ..., K\}$ and L landing spots with equal height H contained within. The LSs and their locations are given by the set

$$\mathcal{L} = \left\{ \left(x_i^l, y_i^l\right),\ i = 1, \ldots, L,\ \ : \left(x_i^l, y_i^l\right) \in \mathcal{G} \right\}.$$

A. UAV Model

The UAV starts its mission from an initial position (x_0, y_0) and is assumed to travel at constant altitude H with a maximum velocity V, i.e. $v(t) \leq V$. The mission is over when the drone's battery is empty defined as time t_f, by which the UAV is supposed to be in the final position (x_f, y_f). During the mission, $t \in [0, t_f]$, the drone's position is given by $x(t)$ and $y(t)$, which are smooth functions of class C^∞ and defined as

$$x : \begin{pmatrix} [0, t_f] \to \mathbb{R} \\ t \to x(t) \end{pmatrix} \qquad y : \begin{pmatrix} [0, t_f] \to \mathbb{R} \\ t \to y(t) \end{pmatrix} \tag{1}$$

subject to

$$x(0) = x_0, \qquad y(0) = y_0 \tag{1a}$$

$$x(t_f) = x_f, \qquad y(t_f) = y_f \tag{1b}$$

The UAV battery's energy content is denoted by

$$b(t) \geq 0, \qquad \forall t \in [0, t_f] \tag{2}$$

during mission time with a full charge when the mission begins, i.e. $b(0) = b_{max}$. Power consumption of the autonomous UAV BS is modeled by a flying and mobility component $p_f(t)$, as well as a communication and computation component $p_c(t)$ which is assumed to be constant during the mission, i.e.

$$p_c(t) = p_c, \qquad \forall t \in [0, t_f]. \tag{3}$$

Energy usage for flying is constant as well, except when the drone has landed in a designated LS, i.e.

$$p_f(t) = \begin{cases} 0, & \forall t\ : (x(t), y(t)) \in \mathcal{L}, \\ p_f, & \text{otherwise.} \end{cases} \tag{4}$$

It follows that the UAV's battery content evolves according to

$$\dot{b}(t) = -p_f(t) - p_c \tag{5}$$

with $\dot{b}(t)$ representing the time derivative of $b(t)$.

B. Communication Channel Model

The communication links between UAV BS and the group of K users are modeled as orthogonal point-to-point channels with log-distance path loss, random small-scale Rayleigh fading and a constant attenuation factor in under non-line-of-sight (NLOS) conditions. The information rate for the k-th user, $k \in \{1, ..., K\}$ located at static position $(a_k, b_k) \in \mathcal{G}$ at ground level is given by

$$R_k(t) = \log_2 \left(1 + \text{SNR}_k(t)\right), \tag{6}$$

where the signal-to-noise ratio (SNR) with transmit power P_k, UAV-user distance $d_k(t)$ and path loss exponent $\alpha = 2$, is defined as

$$\text{SNR}_k(t) = \frac{P_k}{N} \cdot d_k(t)^{-\alpha} \cdot 10^{X_{Rayleigh}/10} \cdot \beta_{shadow}. \tag{7}$$

Small-scale fading was modeled as a Rayleigh distributed random variable $X_{Rayleigh}$ with scaling factor $\sigma = 1$. Attenuation through obstacle obstruction is a discrete factor $\beta_{shadow} = 0.01$ under NLOS conditions, and $\beta_{shadow} = 1$ everywhere else.

Using the described model, the maximization problem can be formulated as

$$\max_{x(t), y(t)} \int_{t=0}^{t_f} \mathrm{E}\left[\sum_{k=1}^{K} R_k(t)\right] dt \tag{8}$$

subject to aforementioned constraints (1a), (1b), (2) and (5).

III. Q-Learning

RL in general and Q-learning in particular, allows an agent to optimize its actions in an environment that can be represented by a Markov decision process (MDP). The learning process constitutes a cycle of interactions between the agent, observing state $s_t \in \mathcal{S}$ and performing an action $a_t \in \mathcal{A}$ at time t, and the environment, which subsequently assigns a reward $r_t \in \mathbb{R}$ to the agent. The state and action space, $\mathcal{S}$ and $\mathcal{A}$ respectively, are part of the problem definition as a finite MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma \rangle$ with state transition probability $\mathcal{P}_a(s, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$, reward function $R_a(s, s')$ and discount factor $\gamma \in [0, 1)$ which controls the importance of future rewards in relation to present reward.

The agent's goal is to learn a behavior or policy that maximizes its received reward, which is given as a distribution over actions given state

$$\pi(a|s) = P\left[a_t = a | s_t = s\right], \qquad \forall a \in \mathcal{A}, s \in \mathcal{S}. \tag{9}$$

To find the optimal policy $\pi^*(a|s)$, Q-learning relies on iteratively improving the state-action value function Q^π which is also called *Q-function*. With respect to a policy π, the Q-function maps each state-action pair to a Q-value representing an expectation of total future reward when following policy π. With the discounted sum of all future rewards at time t called return $R_t \in \mathbb{R}$ and defined as

$$R_t \triangleq \sum_{k=0}^{T-1} \gamma^k r_{t+1+k} \tag{10}$$

and reaching the terminal state at time $t+T$, the Q-function is given as

$$Q^\pi(s,a) = \mathrm{E}_\pi\{R_t \mid s_t = s, a_t = a\}, \quad \forall a \in \mathcal{A}, s \in \mathcal{S}. \tag{11}$$

Without requiring a model and using the observed state transitions and rewards, the Q-function is updated after each transition as follows:

$$Q^\pi(s_t,a_t) \leftarrow Q^\pi(s_t,a_t) + \alpha\left(r_t + \gamma \max_{a'} Q^\pi(s_{t+1},a') - Q^\pi(s_t,a_t)\right) \tag{12}$$

with learning rate $\alpha \in [0,1]$ determining to what extend old information is overridden and discount factor $\gamma \in [0,1)$ balancing the importance of short-term and long-term reward. γ approaching 1 will make the agent focus on gaining long-term reward, whereas choosing $\gamma = 0$ will make it consider only the immediate reward of an action [12].

IV. Neural Network Training and Algorithm

A. Neural Network Approximation of the Q-Function

In traditional Q-learning, the Q-function is usually represented by a multidimensional table that contains one Q-value for each state-action pair. It becomes immediately evident that this is not practical in large state and action spaces as the table size would grow exponentially when adding space dimensions or polynomially when increasing the magnitude of state or action space.

A way out of this dilemma is the use of a NN, the *Q-net*, which approximates the optimal Q-function $Q^*(s,a)$ by a network $Q^\pi(s,a;\theta)$ with parameters θ:

$$Q^\pi(s,a;\theta) \approx Q^*(s,a). \tag{13}$$

The main advantages of this approach include the NN's ability to generalize from few data samples to the whole state space and the NN's higher training data efficiency. A more detailed comparison of table- and NN-based Q-learning for UAV BS trajectory optimization can be found in [7].

Fig. 2 shows the NN architecture used in this work. One state space sample containing drone position, current battery content and landing spot availability forms the input. Two fully connected hidden layers with $n = 100$ units each are followed by the output layer with outputs equal to the cardinality of the drone's action space. All neurons are rectified linear units (ReLU).

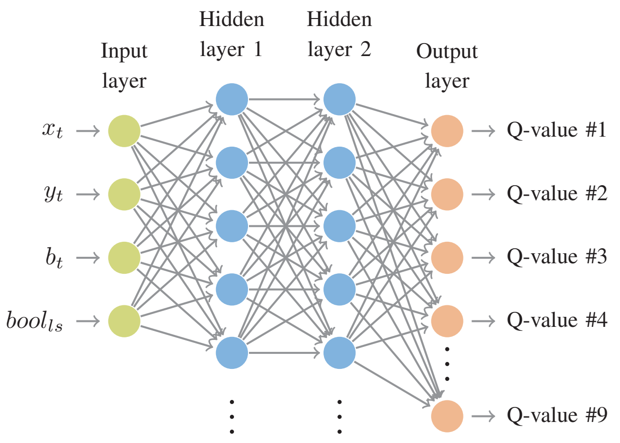

Fig. 2. Neural network architecture consisting of four input and nine output neurons representing the state and action space respectively, as well as two hidden and fully connected layers with $n = 100$ neurons each.

B. Training Process Improvements

It is known that combining Q-learning with a deep NN to represent the Q-function can lead to instability, oscillations and even catastrophic divergence in the training process. Sutton identified three main elements as the cause of such instability calling them the *deadly triad* [12], i.e. function approximation, bootstrapping and off-policy training, which are all present in the Q-net algorithm. Mnih et al. [11] showed that improvements to the training can be made to stabilize the process.

Experience replay [14] is a technique to reduce correlations in the sequence of training data. Each new experience tuple $e = (s_t, a_t, r_t, s_{t+1})$ is stored into a buffer, the replay memory. Instead of directly using the most recent experiences, a mini-batch of temporally uncorrelated data is sampled uniformly from the replay memory and used for training. This increases training data efficiency as samples can be reused multiple times and reduces the variance of updates as the samples' temporal correlation is broken. As those training samples were obtained under different network parameters, the use of replay memory mandates the use of an off-policy learning method such as Q-learning.

The second improvement, *target network separation* [15], addresses the problem of correlated target Q-value y_i and Q-value estimate $Q(s_i, a_i; \theta)$ inside the loss function as described in line 13 of Algorithm 1 below. Both values are taken from the same NN which can lead to oscillations or divergence in the policy. By cloning the primary network periodically and generating target values exclusively from this separate target network, a delay is added between the time an update is made at the time the update affects the targets y_i, making divergence and oscillations less likely [11].

C. Training Algorithm

In the following, the NN training algorithm is described in more detail.

Algorithm 1 Q-network training [11]

```
    Initialize replay memory $M$ to size $D$
    Initialize primary network $Q$ with random parameters $\theta$
    Initialize target network $\hat{Q}$ with random parameters $\hat{\theta}$
1:  for $n = 0$ to $N_{max}$ do
2:      Initialize state $s_0 = (x_0, y_0, b_{max}, bool_{ls})$, $t = 0$
3:      if ($n \bmod N_{target} = 0$) then
4:          $\hat{\theta} \leftarrow \theta$
5:      end if
6:      while $b \geq 0$ do
7:          $a_t = \begin{cases} \text{randomly select from } \mathcal{A}, & \text{with probability } \epsilon \\ \arg\max_{a'} Q(s_t, a'; \theta), & \text{otherwise} \end{cases}$
8:          Observe $r_t$, $s_{t+1}$
9:          Store $e = (s_t, a_t, r_t, s_{t+1})$ in $M$
10:         for $i = 0$ to $m$ do
11:             Sample $(s_i, a_i, r_i, s_{i+1})$ uniformly from $M$
12:             $y_i = \begin{cases} r_i, & \text{if terminal} \\ r_i + \gamma \max_{a'} \hat{Q}(s_{i+1}, a'; \hat{\theta}), & \text{otherwise} \end{cases}$
13:             Compute $L_i(\theta, \hat{\theta}) = \mathrm{E}\left[(y_i - Q(s_i, a_i; \theta))^2\right]$
14:         end for
15:         $\theta \leftarrow \theta - \eta \cdot \frac{1}{m} \nabla_\theta \sum_{i=1}^{m} L_i(\theta, \hat{\theta})$
16:         $t = t + 1$
17:     end while
18:     $\epsilon \leftarrow \epsilon_{final} + (\epsilon_{start} - \epsilon_{final}) e^{-\lambda n}$
19: end for
```

After initialization of replay memory buffer and network parameters, a new learning episode is started by resetting the time index and the drone's position, as well as the drone's battery (line 2). Every N_{target} episodes, the target network parameters $\hat{\theta}$ are updated (line 4).

As long as there is energy left in the UAV's battery, the mission continues and the agent makes a movement decision according to the ϵ-greedy policy. With probability ϵ, an action is randomly selected from the action space $\mathcal{A}$ and otherwise the action that maximizes the Q-function in the current state. Subsequently, the environment assigns a reward r_t and propagates the UAV to the next state s_{t+1}. The new experience tuple is saved in the replay memory (line 9).

To train the Q-net, a minibatch of m experiences is sampled uniformly from the replay buffer, the target value y_i is set using the target network and the loss computed (lines 10-14). Using SGD or a derivative thereof, the primary network parameters θ are updated with learning rate η (line 15). After the battery is empty and the mission is over, the probability for random exploration of the state space is exponentially decayed with decay constant λ (line 18). Algorithm 1 terminates when the final learning episode $n = N_{max}$ is reached.

V. Simulation

As depicted in Fig. 3, the algorithm is tested in a grid world of size $1000\,\mathrm{m} \times 1000\,\mathrm{m}$ discretized by $100\,\mathrm{m}$ steps (121 unique geometric positions). The UAV's start and final position are at the origin and the upper right corner, respectively. The UAV serves $K = 10$ users. While Fig. 3 illustrates the scenario with $L = 1$ LS, Fig. 4 depicts the $L = 2$ LS scenario.

In addition to landing on a LS, the action space of the UAV is limited to 8 movement direction due to the geometric restrictions. Possible control actions are given by

$$v \in \left\{ \begin{bmatrix} 0\,\mathrm{m\,s^{-1}} \\ 0 \end{bmatrix}, \begin{bmatrix} 12.5\,\mathrm{m\,s^{-1}} \\ \phi \end{bmatrix}, \begin{bmatrix} 17.7\,\mathrm{m\,s^{-1}} \\ \phi + \frac{\pi}{4} \end{bmatrix} \right\}$$

with $\phi \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. The UAV is assumed to travel at a constant altitude of $H = 40\,\mathrm{m}$. The drone's battery content is $b_{max} = 31$ Wh when fully charged. Power consumption for flying and communication is assumed to be $p_f = 400$ W and $p_c = 40$ W, respectively. These values reflect our experiences from real-world UAV BS experiments at Eurecom [10].

The basis of the reward signal is formed by the instantaneous sum information rate as computed in Equation (6). Additional punishments (negative reward) are given to the drone for stepping out of the 11 by 11 grid world, as well as for failing to return to the final position before the battery runs out and the mission is over. For discovering and landing in a LS for the first time, a one-off "discovery" reward is given to the agent. As the numerical value of these punishments and rewards can be chosen freely by the environment designer, it is reasonable to fix them to values in the same range as the expected sum rate, the reward signal's main component.

VI. Results

Fig. 3 depicts the final learned trajectories when applying Algorithm 1 to a scenario similar to the one investigated in [6], where dynamic programming (DP) was used to obtain a model-based solution. Two trajectories are shown for cell-edge SNRs of 10dB and -15dB. The cell-edge SNR is defined as the SNR of the radio link between the UAV at center position (500m, 500m) and a user maximally far apart, e.g. at (0m, 1000m). For a low cell-edge SNR, the drone ignores the LS and instead flies around the maximum sum rate point near the user cluster in the lower right corner and returns to the final position in time before its battery runs out. Under high cell-edge SNR conditions (10dB), the UAV BS learns to obtain an overall higher sum rate result by landing in the LS and extending the mission duration by conserving energy. The results obtained with Algorithm 1 mirror the DP model-based results, which are known to be optimal [6].

In addition to the environmental parameters described in Section V, the second scenario includes an obstacle that causes shadowing and obstructs the LOS connection to some users. The shadowed areas are depicted in Fig. 4 as gray, where the darker regions are shadowed from multiple users. The right LS is in the shadow of the obstacle whereas the left LS provides a LOS connection to all 10 users.

Fig. 4 shows two results for high shadowing loss ($\beta = 0.01$) and low shadowing loss ($\beta = 0.1$), both under high cell-edge SNR conditions (10dB). With low shadowing, the drone stays on the direct line between start and final position until reaching the LS where it lands and conserves energy. With the minimum amount of energy left in the battery that is required to reach

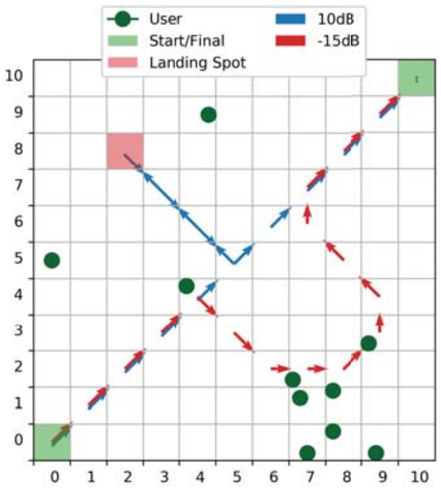

Fig. 3. Final trajectories for cell-edge SNRs of 10dB and -15dB.

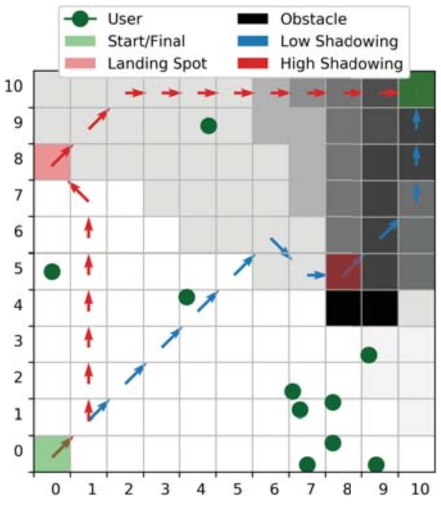

Fig. 4. Final trajectories for high and low shadowing loss factor.

the final position, it restarts from the LS to arrive at the final position in time.

In contrast, higher shadowing loss leads the agent to realize that the left LS, despite being far away from most users and requiring more energy to reach, leads to a better overall sum rate result in the long run. Even under challenging conditions with random small-scale fading and shadowing obstacles in the environment, the Q-net agent is able to discriminate between multiple LSs and different channel conditions to achieve the best long-term result. Fig. 5 shows the overall collected data per training episode (or completed mission) during the learning process. The training converges for both shadowing conditions to a stable solution at about $n = 32,000$ episodes.

VII. Conclusion

We have introduced a Q-learning system that trains a NN to make movement decisions for an autonomous UAV BS under an energy constraint with the help of LSs. In contrast to previous works [6], the presented system can utilize LSs efficiently to extend mission duration and maximize the sum rate of the transmission without a model or any prior information about the environment. The training procedure was shown to adapt effectively to complex environmental effects like small-scale fading and obstacle shadowing.

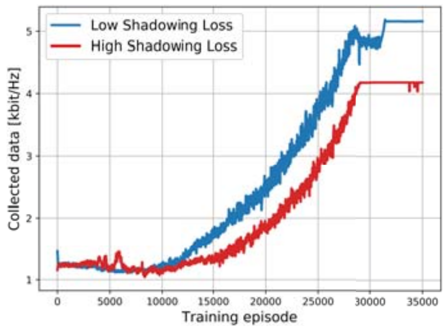

Fig. 5. Collected data per training episode for trajectories in Fig. 4.

References

[1] P. E. Ross, "Iceland's consumers try drone delivery - [news]," *IEEE Spectrum*, vol. 55, no. 10, pp. 12–13, Oct 2018.

[2] J. Kosmerl and A. Vilhar, "Base stations placement optimization in wireless networks for emergency communications," in *IEEE International Conference on Communications Workshops (ICC)*, 2014, pp. 200–205.

[3] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Optimal transport theory for power-efficient deployment of unmanned aerial vehicles," in *IEEE International Conference on Communications (ICC)*, 2016.

[4] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[5] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, 2018.

[6] R. Gangula, D. Gesbert, D.-F. Külzer, and J. M. Franceschi Quintero, "A landing spot approach to enhancing the performance of UAV-aided wireless networks," in *IEEE Workshop on Integrating UAVs into 5G, International Conference on Communications (ICC)*, 2018.

[7] H. Bayerlein, P. de Kerret, and D. Gesbert, "Trajectory optimization for autonomous flying base station via reinforcement learning," in *19th IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018.

[8] A. Fotouhi *et al.*, "Survey on UAV cellular communications: Practical aspects, standardization advancements, regulation, and security challenges," *arXiv preprint arXiv:1809.01752*, 2018.

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

[10] (2017) Eurecom - autonomous aerial cellular relaying robots (video). [Online]. Available: https://youtu.be/GI_lOsg_qmQ

[11] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[12] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 2nd ed. MIT Press, 2018.

[13] U. Challita, W. Saad, and C. Bettstetter, "Cellular-connected UAVs over 5G: Deep reinforcement learning for interference management," *arXiv preprint arXiv:1801.05500*, 2018.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[15] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.