Optimal Trajectory of Autonomous Flying Base Stations via Reinforcement Learning

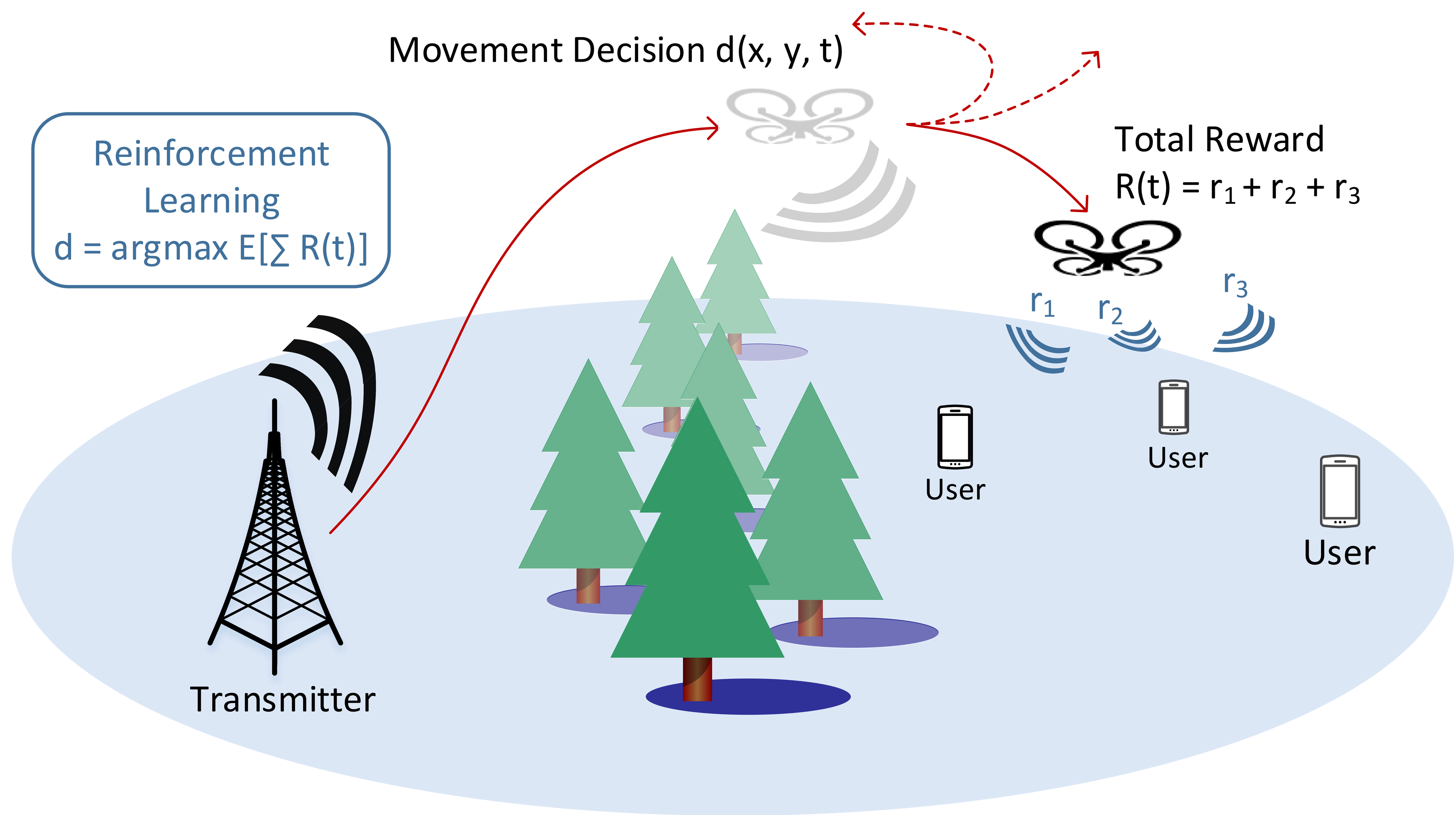
Harald Bayerlein, Paul de Kerret and David Gesbert

Communication Systems Department, Eurecom

Autonomous UAV Base Station [1][2]

- Quadcopter UAV acts as a relay between users and a stationary transmitter
- Useful for dynamic network deployment and fast response to varying demand, e.g. to sustain communications ability in disaster situations
- System performance mainly depends on UAV trajectory

➡ Trajectory planning must optimize link quality while observing constraint on flying time!



System Model

- UAV position with constant altitude and constant velocity V, flying time T:

$$x: \begin{pmatrix} [0,T] \to \mathbb{R} \\ t \to x(t) \end{pmatrix} \quad y: \begin{pmatrix} [0,T] \to \mathbb{R} \\ t \to y(t) \end{pmatrix}$$

$$\text{s.t.} \quad x(0) = x_0, \quad y(0) = y_0$$
$$x(T) = x_f, \quad y(T) = y_f$$

- Pathloss:

$$L = d_k(t)^{-\alpha} \cdot 10^{X_{Rayleigh}/10}$$

$$d_k(t) = \sqrt{H^2 + (x(t) - a_k)^2 + (y(t) - b_k)^2}$$

- Orthogonal point-to-point channel with information rate for k-th user

$$R_k(t) = \log_2\left(1 + \frac{P}{N} \cdot L\right)$$

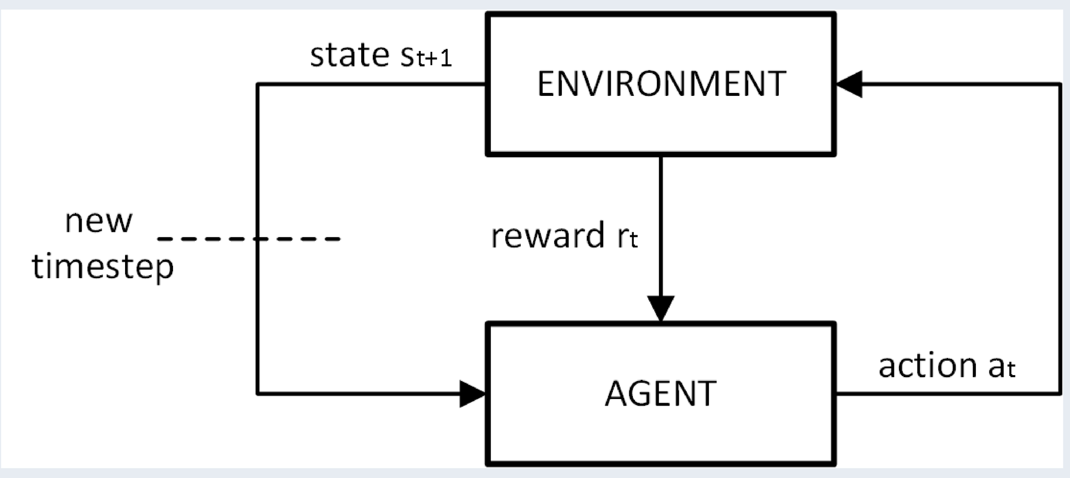
➡ Maximization problem over K users:

$$\max_{x(t),y(t)} \int_{t=0}^{T} \sum_{k=1}^{K} R_k(t)dt$$

➡ Use Reinforcement Learning to learn optimal strategy

Reinforcement Learning [4]

Main idea: an *agent* in an environment takes *actions* and tries to maximize the *reward* it perceives subsequently



- Modelled as finite MDP $\langle S, A, P, R, \gamma \rangle$
- *Policy*

$$\pi(a|s) = \mathcal{P}[A_t = a | S_t = s]$$

- *Action-value* function

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\}$$

➡ Optimal policy $\pi^*(a|s) = \text{argmax}_a Q^{\pi^*}(s,a)$

Q-Learning [3]

Bellman Optimality Condition:

$$Q^{\pi^*}(s_t, a_t) = r_t^* + \gamma \max_a Q^{\pi^*}(s_{t+1}, a)$$

➡ Solve Bellman Optimality Equation iteratively

- $Q^{\pi}(s_t, a_t)$ is updated after carrying out action a_t and receiving reward r_t for it

$$Q^{\pi}(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha\left(r_t + \gamma \max_a Q^{\pi}(s_{t+1}, a) - Q^{\pi}(s_t, a_t)\right)$$

- Discount factor $\gamma \in [0,1)$ balances short-term/ long-term reward
- Learning rate $\alpha \in [0,1]$ controls to what extend old information is overridden
- Q-Learning finds an optimal policy for any finite MDP

Application of Q-Learning to Path Planning

Simulation Parameters

- State variables: (x, y, t)
- Actions $a \in [up, right, down, left]$
- $x_0 = x_f$ and $y_0 = y_f$
- Maximum flying time $T = 50$
- 15×15 grid, two users and one 2×4 obstacle causing shadowing
- Policy: ϵ-greedy with ϵ exponentially decreasing with decay constant 10^{-5}
- Negative reward for stepping out of the 15x15 grid and for activation of "force return"
- Number of learning episodes: $1 \cdot 10^6$
- Learning rate $\alpha = 0.3$, discount factor $\gamma = 0.99$

Learning Results

➡ Agent finds maximum cumulative rate point

➡ Minimum shadowing trajectory is learned

➡ Agent learns to return autonomously

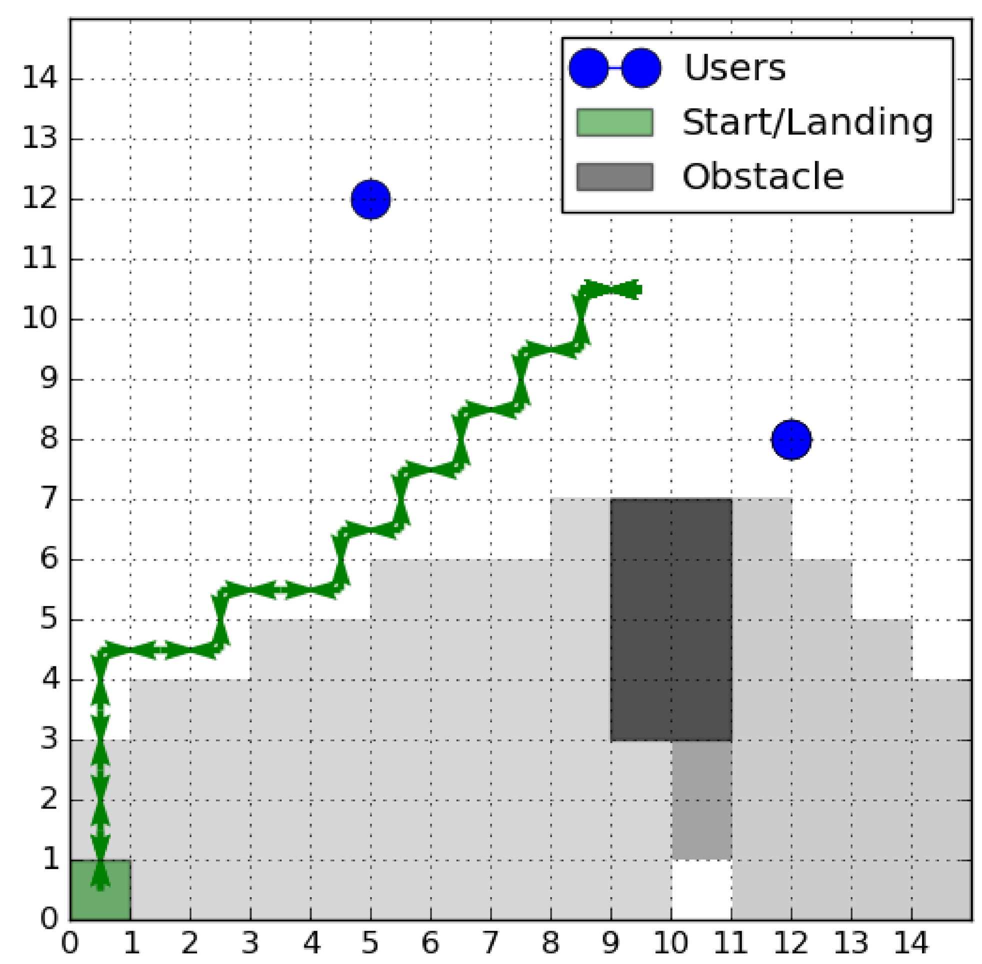


Figure: Final learned trajectory after 1 Million episodes

Extensions

- Consideration of relaying function
- Random fading and complex topology
- Large state and action spaces

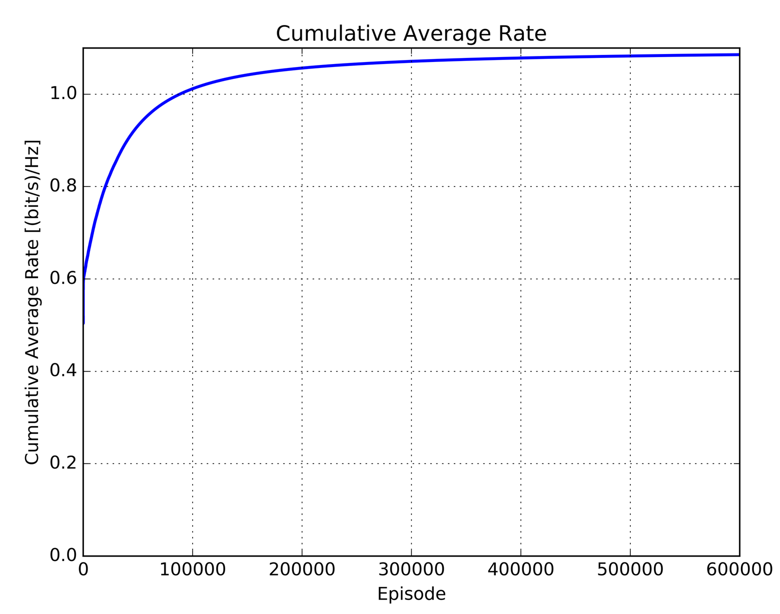


Figure: Cumulative average rate over episode

References

[1] R. Gangula, P. de Kerret, O. Esrafilian and D.Gesbert, "Trajectory Optimization for Mobile Access Point", ASILOMAR, Pacific Grove, CA, 2017.

[2] J. Chen and D. Gesbert, "Optimal positioning of flying relays for wireless networks: A LOS map approach," 2017 IEEE ICC, Paris, 2017, pp. 1-6.

[3] C. J. C. H. Watkins and P. Dayan, "Q-learning", *Machine Learning*, vol. 8, no. 3-4, 1992.

[4] R. S. Sutton and A. G. Barto, "Introduction to Reinforcement Learning", 2nd ed. Cambridge, Massachusetts: MIT Press, 2017.