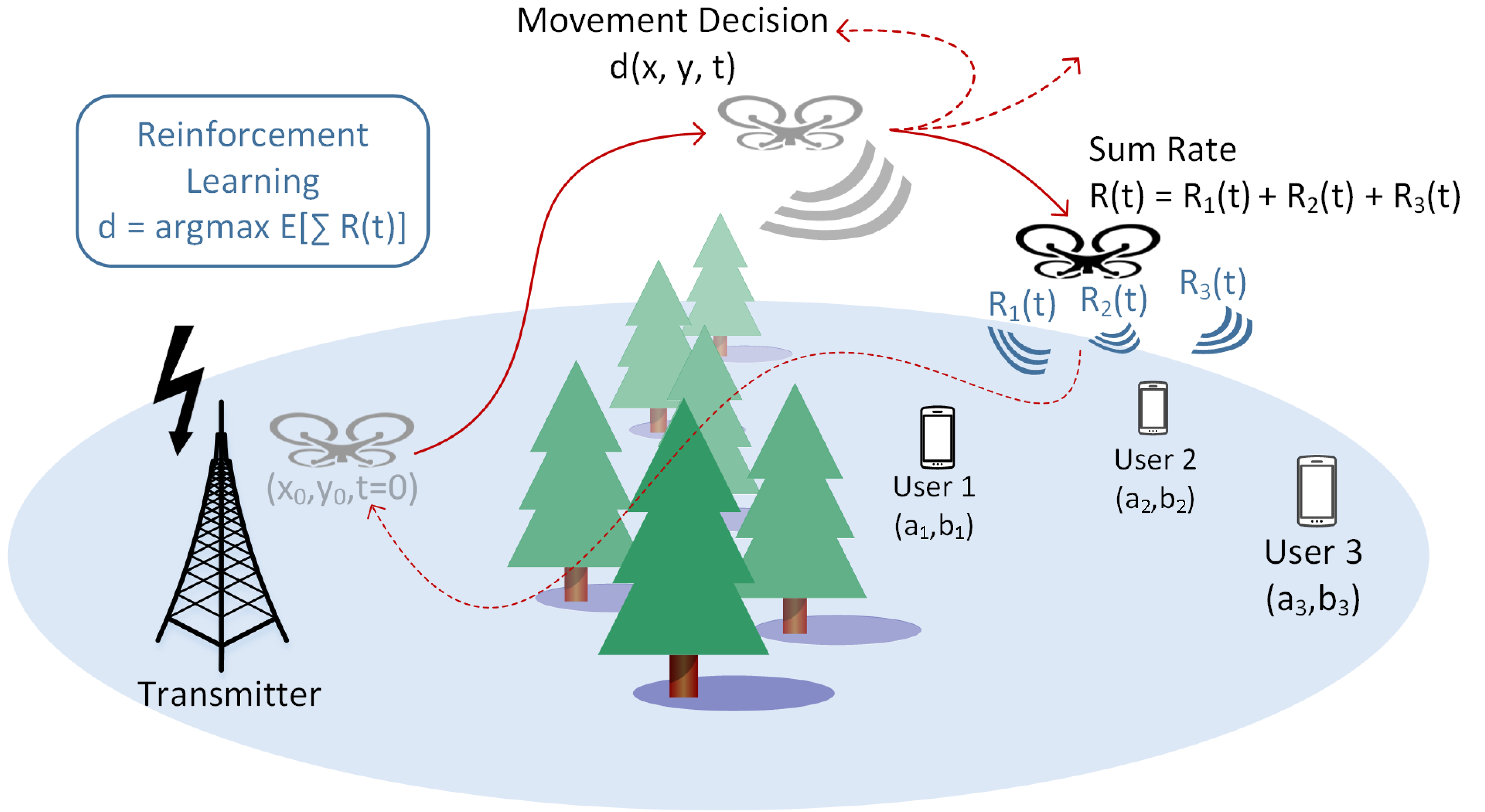
Trajectory Optimization for Autonomous Flying Base Station via Reinforcement Learning

Harald Bayerlein, Paul de Kerret and David Gesbert
Communication Systems Department, EURECOM

Autonomous UAV Base Station

- 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. 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:

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

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

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

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

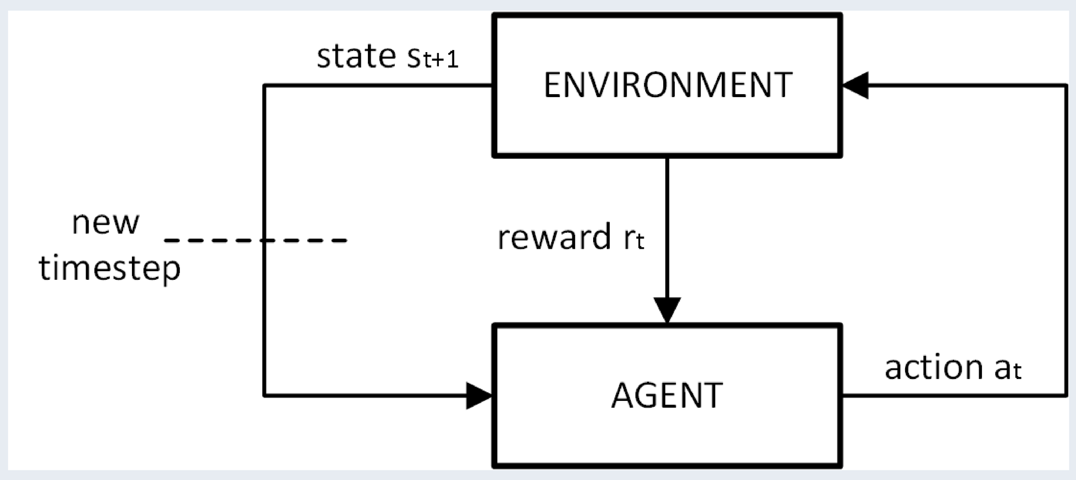
➡ Maximization problem over K users:

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

➡ Use Reinforcement Learning to learn optimal strategy

Reinforcement Learning

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*

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

- *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 [1]

Bellman Optimality Condition:

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

➡ Solve Bellman 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

➡ Compare Q-function approximators: lookup table (Q-table) and neural network (Q-net)

Application of Q-Learning to Trajectory Planning

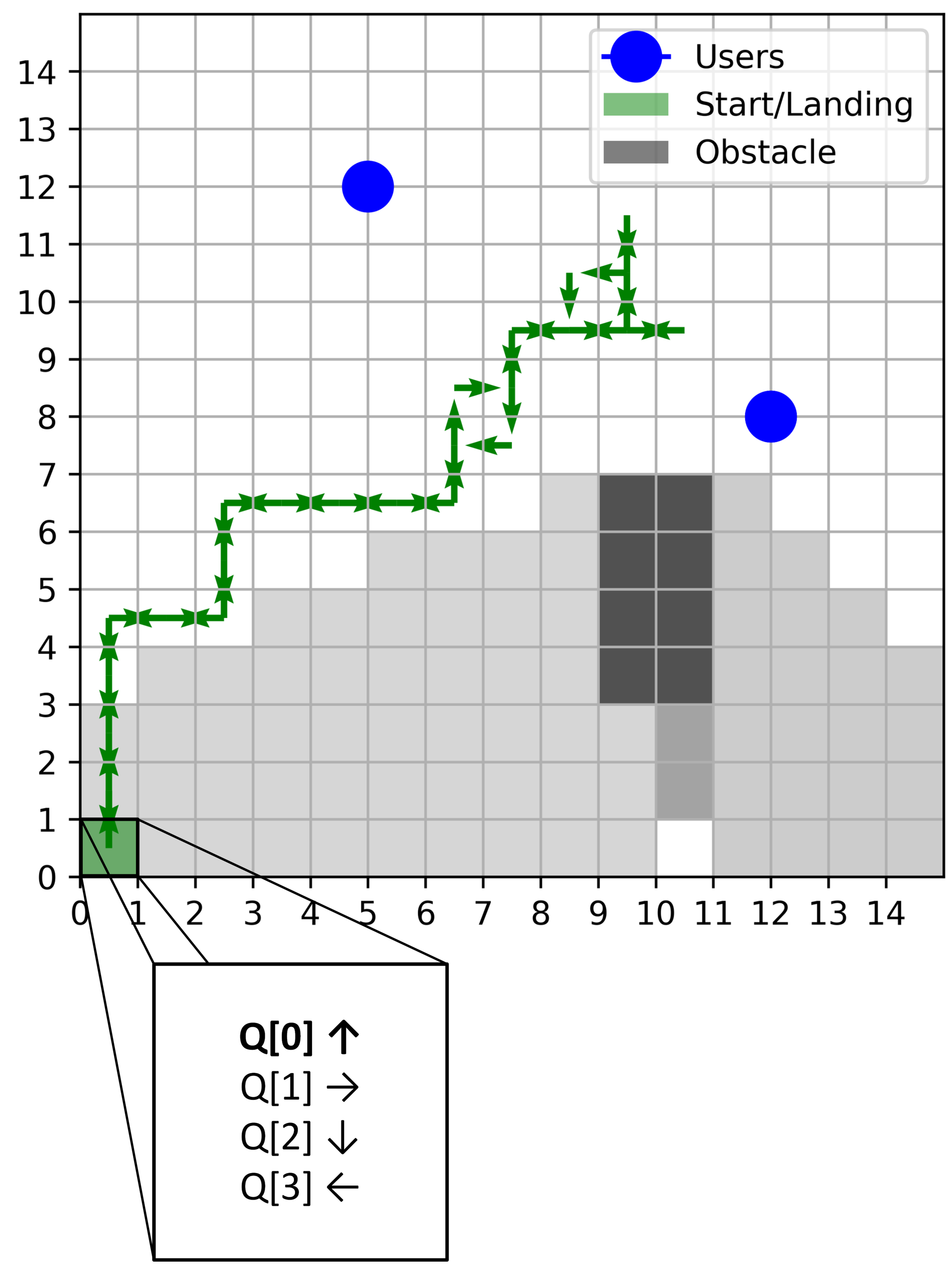


Figure: Final trajectory after 800,000 training episodes for the Q-table approach, whereas 27,000 suffice for Q-net

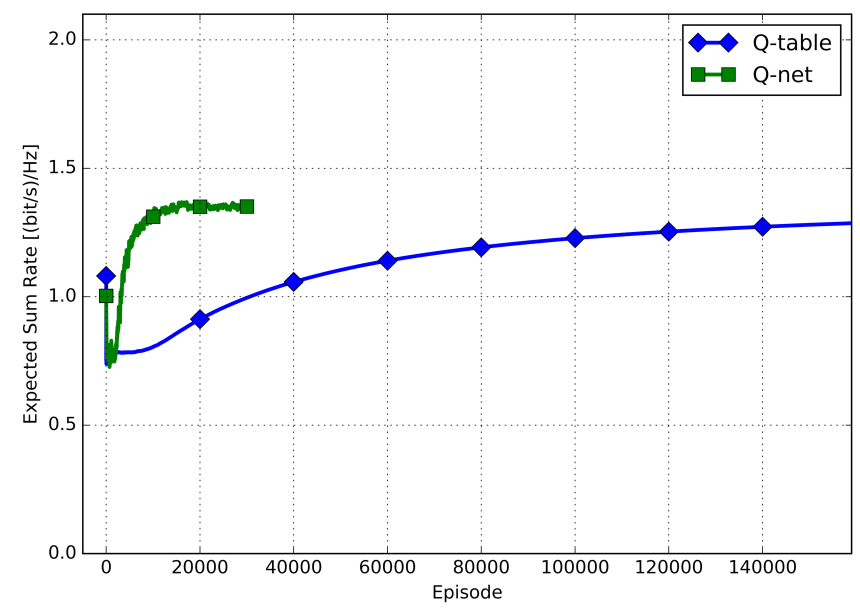


Figure: Expected sum rate over training time

Learning Results

➡ Agent finds maximum cumulative rate point
➡ Minimum shadowing trajectory is learned
➡ Agent learns to return autonomously

Extensions

- Consideration of relaying function
- Dynamically changing environment
- Trajectory energy efficiency

Q-Learning with NN: Q-Net [2]

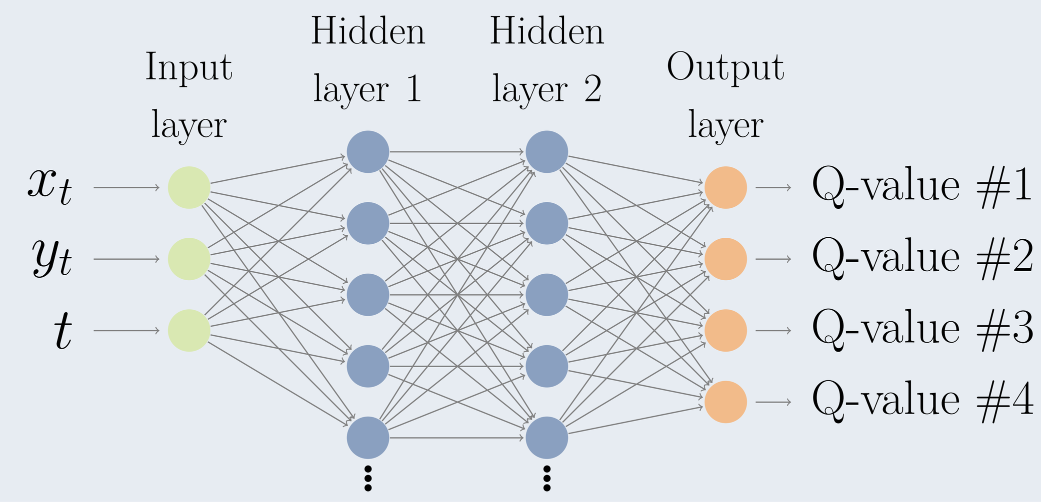
- Use neural network (NN) with parameters θ to approximate Q-function:

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

- Minimize loss function at each iteration i:

$$L_i(\theta_i) = E\Big[\big(\overbrace{r_t + \gamma \cdot \max_{a'} Q(s_{t+1},a';\theta_i)}^{\text{target Q-value}} - Q(s,a:\theta_i)\big)^2\Big]$$

- Neural network model:



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

[2] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, no. 7540, 2015.