Appendix:

I. Environment

A. General parameters

- 100 by 100 meters squared area.
- 1050 Users with three clusters
- The number of ABSes is 2, 4, 8.

B. User distribution

- The users are distributed around a local cluster.
- The center of the cluster is determined at random.
- It can take positions from 30 to 70, both in x and y coordinates.
- The users are scattered randomly with normal distribution around the cluster center, with mean $\mu = 0$ and standard deviation $\sigma = 20$.
- An example of the user distribution can be seen in Fig. 1. Total three clusters are simulated.

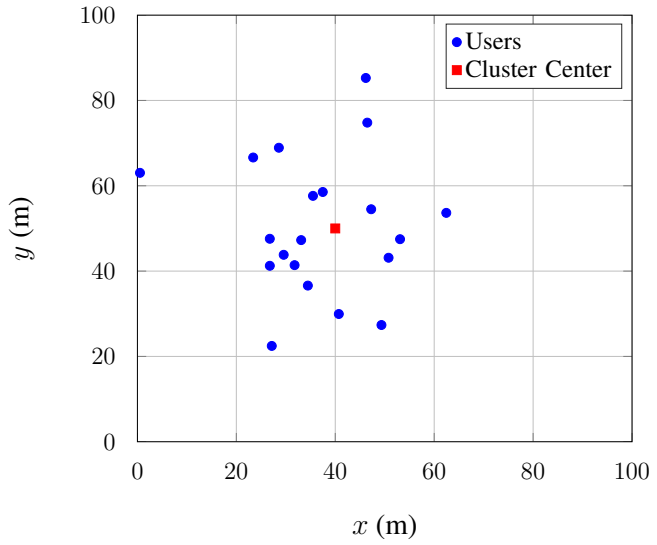

Fig. 1. Example of user distribution for one cluster in one training session.

C. Airbone Base Stations (ABSes)

- The ABSes are assumed to have ideal communication among themselves.
- The ABSes share their positions and number of users allocated with each other.
- Each ABS has a directional antenna with a main lobe with an aperture angle of $\theta = 60$ degrees. The antenna is pointing downwards. An illustration of the directivity angle is presented in Fig. 2
- There is no limit on the number of users that each ABSes can allocate.
- Each ABS is assumed to be flying at a fixed height $h_{\mathrm{d}} = 30$ meters.

II. States (for two ABSes Cases)

- The states are the positions of both drones.
- The environment is discretized into 121 possible positions for each ABS (steps of 10 meters).
- Drones cannot assume the same position at the same time.
- The total number of states is therefore $2 \times \binom{121}{2} = 14520$.

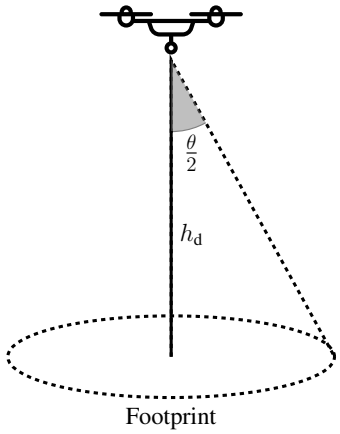

Fig. 2. Illustration of the coverage radius of a ABSes flying at h_{d} meters and with directivity angle of θ.

III. Actions

- The possible actions are to move ± 1 step in x or y.
- If one ABS would move to the same position as the other (e.g˙chooses the action to move right when the other ABS is one step to its right), it does not move.
- If a ABS would move out of the grid (e.g˙chooses to move right when at x coordinate 100), it does not move.
- If a state has not been explored, the action is also chosen at random, in order to avoid bias.
- Otherwise, actions are deterministic.

IV. Policy

- The policy is ε-greedy.Meaning that each ABS chooses a random action with probability ε and maxQ with probability 1 − ε.

V. Reward

- The reward is the total number of users allocated by all ABSes.

A. User Association

- A user associates with a ABSes if its received SINR is above a threshold of 40 dB.
- The SINR for the link between user u and ABS n is calculated according to

$$SINR_{n,u} = \frac{RSRP_{n,u}}{N + \sum_{\forall i \neq n} RSRP_{i,u}}. \quad (1)$$

- The RSRP for the link between user u and ABS n is calculated according to the free space path loss, as

$$RSRP_{n,u} = \frac{P_{\mathrm{t}}}{\frac{16(\pi f_{\mathrm{c}} d)^2}{c^2}}, \quad (2)$$

where c is the speed of light in meters per second, P_{t} is the ABSes transmit power in Watts and f_{c} is the carrier frequency in Hz.

- Any user outside the main lobe is considered to receive 0 W of power from the ABS.

VI. Training

- The typical training session is comprised of 10 episodes of 2000 iterations each. At DQN we use 1000 episodes •

The state of the ABSes is set randomly at the beginning of each episode.

- ε decays exponentially with the episode number, according to:

$$\varepsilon_j = \mathrm{e}^{-j/20}, \tag{3}$$

where j is the episode number and e is Euler's constant.

A. Algorithm (SARSA)

The update strategy for SARSA is expressed as

$$\begin{aligned} Q(s_t, a_t, \delta) = Q(s_t, a_t, \delta) + \\ \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}, \delta) - Q(s_t, a_t, \delta)), \end{aligned} \tag{4}$$

where α is the learning rate and γ is the discount factor. A pseudo code of the implemented solution is presented in Algorithm 1.

Algorithm 1: SARSA implementation

```
Initializations
for Every episode j do
    s_1 ← random(1, 14520)
    for Every iteration t do
        for Every ABSes δ do
            a_t ← chooseAction(Q_{s_t,*}, ε_j, δ)
            s_{t+1} ← takeAction(s_t, a_t, δ)
            a_{t+1} ← chooseAction(Q_{s_{t+1},*}, ε_j, δ)
            r_t ← computeReward(s_{t+1})
            Q(s_t, a_t, d) ← updateQ()
            s_t ← s_{t+1}
        end
    end
end
```

VII. Results

The average reward per episode is shown in Fig. 3.

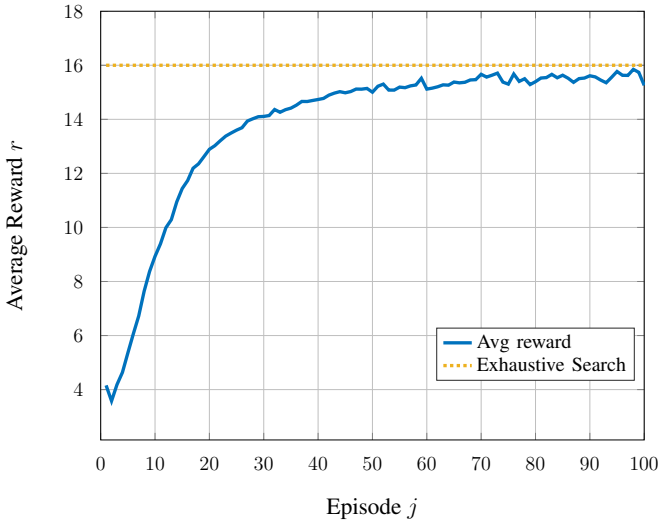

Fig. 3. Average reward per episode considering for this training session