I. Environment

A. General parameters

- 100 by 100 meters squared area.
- 1050 Users.
- Each User is assumed to hold their device at a fixed height $h_{\mathrm{u}} = 1.5$ meter.

B. User distribution

- The users are distributed under three 2-dimensional Gaussian distillation layers and one uniform distribution layer
- 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 300, 250 and 200 MSs follows a 10, 7 and 6 meters standard deviation respectively.
- 300 MSs follow a uniform distribution

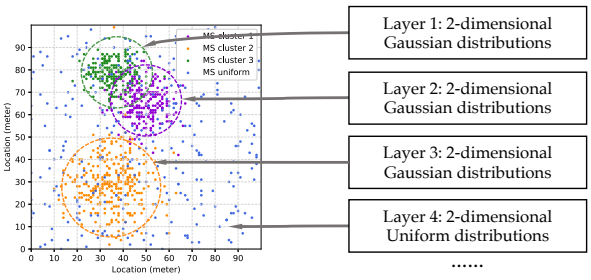

Fig. 1. Environment construction

C. Drones

- The drones are assumed to have ideal communication among themselves.
- The drones share their positions and number of users allocated with each other.
- Each drone 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 drone can allocate.
- Each drone is assumed to be flying at a fixed height $h_{\mathrm{d}} = 30$ meters.

II. States

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

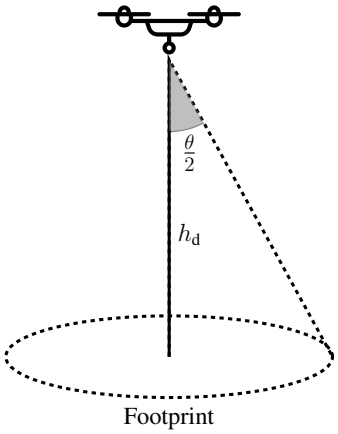

Fig. 2. Illustration of the coverage radius of a drone flying at h_{d} meters and with directivity angle of θ.

III. Actions

- The possible actions are to move ± 1 or ± 0 (stay) step in x or y.
- If one drone would move to the same position as the other (e.gċhooses the action to move right when the other drone is one step to its right), it does not move.
- If a drone would move out of the grid (e.gċhooses 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 drone chooses a random action with probability ε and $\max(Q_{s_{t+1},*})$ with probability $1 - \varepsilon$.

V. Reward

- The reward is the total number of users allocated by both drones.

A. User Association

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

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

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

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

where c is the speed of light in meters per second, P_{t} is the drone 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 drone.

VI. Training

- The training session is comprised of 100 episodes of 20,000 iterations each.
- The state of the drones 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

The update strategy for DQN is expressed as,

$$L(\theta) = E\left[(\text{TargetQ} - Q(s,a;\theta))^2\right] \tag{4}$$

where TargetQ is present as,

$$\text{TargetQ} = r + \gamma \max_{a'} Q\left(s', a'; \theta\right) \tag{5}$$

where α is the learning rate and γ is the discount factor. A pseudo code of the implemented solution is presented as follow.

Algorithm 1: DQN implementation

```
Initialization
for every episode j do
    s_1 ← random
    for Every iteration t do
        for Every drone δ do
            a_t ← max_a Q(φ(s_t), a; θ) with probability ϵ
             select a random action
            r_t, x_{t+1} ← based on the action a_t
            φ_{t+1} = φ(s_{t+1}) ← s_{t+1} = (s_t, a_t, x_{t+1})
            D ← Add to dataset D + (φ_t, a_t, r_t, φ_{t+1})
            evaQ_{t+1} ← evaQ learn from data in D
            s_t ← s_{t+1}
        end for
        Q ← fine-tune from evaQ for every n steps
    end for
end for
```

VII. Results

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

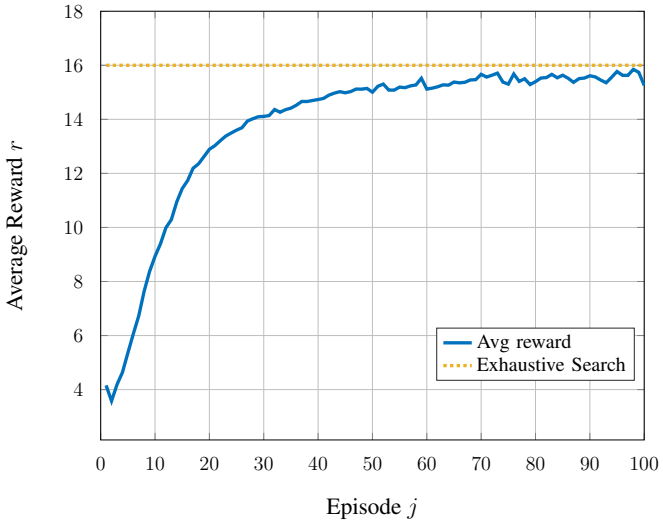

Fig. 3. Average reward per episode considering for this training session