Joint Power Control, Pilot Assignment, User Association and Flight Control for Massive MIMO Self-Organizing Drones using Reinforcement Learning

Gabriel Skidmore
Department of Computational Electrical and Computer Engineering
Miami University
Oxford, United States
Email: skidmogm@miamioh.edu

Abstract—**Improving spectral efficiency is becoming increasingly important in mobile communications to keep up with the ever-increasing amount of data traffic coming from video streaming, Internet of Things, intelligent transportation systems, and augmented and virtual reality. In this work, a deep reinforcement learning algorithm (Deep Q-Learning) is implemented to maximize the sum spectral efficiency of ground users using Unmanned Aerial Vehicles (UAVs) as agents. The agents and environment are created by using OpenAI's Gym library to create a custom implementation of the agent, reward function, and environment. The problem is then relaxed by assigning users to UAVs that lead to the highest Single-Input Single-Output (SISO) Signal to Interference plus Noise Ratio (SINR) and allowing the UAVs to assign multiple pilot signals to ground users. Lastly, the implementation of the algorithm is compared to a convex relaxed version of the original reward function.**

Keywords- Wireless Drone Networking; Massive MIMO; Deep Q-Learning; Reinforcement Learning; Nonconvex Optimization.

I. Introduction

With recent technological innovations in telecommunications and the exponential increase in wireless data traffic from video streaming, Internet of Things (IoT), augmented reality, and surveillance, Unmanned Aerial Vehicles (UAVs) are being considered for use as Mobile Base stations (BSs) with massive Multiple-Input Multiple-Output (MIMO) networking capabilities [1], [2]. By enabling UAVs with Massive MIMO, different applications can be achieved, such as spectrum sharing through beamforming, unlicensed spectrum sharing with redeployable aerial drone base stations, secure wireless networking in congested environments through directional communication, and edge computing. Having UAVs act as massive MIMO mobile BSs is done to provide spectral efficient wireless networking for ground users. UAV BSs maximize spectral efficiency by changing location, controlling ground user transmit power, and assigning pilot sequence to users that maximizes the sum spectral efficiency of all users. It is worth noting that there are many factors that go into the operation of a drone, and they are impacted by many factors, such as weight, battery, energy consumption, and flight trajectory [4]. There have been many new approaches to extend the operation of a drone, such as battery swapping, using hybrid fuel cells and batteries, and solar cells for an extra power source [5]. These advancements are not the focus of this article which lets this article focus on the networking between ground users and the UAV BSs.

Massive MIMO adds the ability for wireless links to achieve higher throughput without the need of adding more BSs or increasing bandwidth. At the same time, UAVs have the advantage of being able to change location to follow demand. To maximize throughput, a massive MIMO UAV needs to find the best location and best pilot sequences to assign to users that minimizes interference between users. When multiple UAV BSs are used, each UAV BS has the added constraint of minimizing their connected users' interference with users connected to other BSs. It is reasonable to mount massive MIMO on an UAV BS because massive MIMO can reach smaller form factors if each ground node has distinct spatial channel characteristics [3]. For example, in a 2 GHz frequency band, for 100 dual-polarized antennas, the antenna array only requires 0.75 x 0.75 meters of space.

Controlling ground users' transmit power and pilot sequence allocation and the UAV BSs' movement is important in order to maximize ground users' sum spectral efficiency. However, this is a difficult problem because ground users can only be connected to one drone and can only be assigned one pilot sequence. This makes it a Mixed Integer Nonlinear Programming (MINLP) problem, which is generally NP hard. This means there are no known globally optimal solutions with polynomial computational complexity.

In recent literature, there are many solutions that do not use reinforcement learning. [7], [8] look to minimize user outage, [9] maximizes user spectral efficiency, [10], [11] maximize throughput, and [12] maximizes the Received Signal Strength Indicator (RSSI). There are also solutions that do use reinforcement learning. [13], [14] minimize user outage, [15], [16] maximize throughput, and [17] maximizes the RSSI. The main goal of these approaches is to maximize user experience. Some of these solutions, [7], [8] and [12] focus on assisting a central terrestrial BS, [9] - [11], [13] - [15] focus on using UAVs, Long-Term Evolution (LTE) small cells, and access points as the BSs, and [17] only focuses on efficiently allocating spectrum for one cognitive radio network.

While many of these solutions are used in real systems, there are some drawbacks. First, LTE small cells, access points, and hotspots are not able to achieve the same level of spectral efficiency that massive MIMO can reach. Second, using terrestrial BSs is too expensive if there is only a short term spike in user demand. Finally, the last drawback is that solutions only meant to allocate spectrum to one radio network might not perform well if allocating resources to multiple radio networks at the same time.

To deal with these drawbacks, UAVs are used as mobile BSs mounted with massive MIMO antenna arrays. To maximize spectral efficiency, the locations of all the UAVs, the pilot signals assigned to the ground users and their transmit power are all maximized using Deep Q-Learning. Since this is a MINLP problem, the problem is relaxed by allowing the UAVs to assign multiple pilot sequences to ground users and users are assigned to UAVs that lead to the highest SISO SINR. These two steps remove the binary constraints in pilot assignment and user association. Results so far show that with two UAVs and two users, the UAVs are able to provide 95% of the optimum sum spectral efficiency.

The rest of the paper is organized as follows. Related work and background is reviewed in Section II. Problem formulation and implementation details are described in Section III. Sections IV goes over simulation results and includes a discussion of future work. Lastly, conclusions are drawn in Section V.

II. Related Work

A. Unmanned Aerial Vehicle Base Stations

An unmanned aerial vehicle network consists of an area with one or more UAVs mounted with BSs and multiple users or user equipment that need to connect to the network. The UAVs can change their location to provide better coverage, throughput, or spectral efficiency for users. This system could be paired with already existing ground BSs or be part of a public safety network after the occurrence of a natural disaster.

To see the advantages mobile BSs have over traditional static BSs, [10] checks the practical limits of UAVs in a cellular network and comes up with a mobility control algorithm to maximize the spectral efficiency at different UAV speeds. There are different scenarios where mobile UAV BS positioning is important. When using UAVs as mobile hotspots, [9] positions UAVs to minimize the distance between UAVs and users to boost packet throughput and the average packet throughput. UAV BSs can also be used for IoT networking where [12] used a drone mounted with a Raspberry Pi to create an IoT mesh to maximize the Received Signal Strength Indicator (RSSI). Drones can also be paired with traditional terrestrial BSs to further improve coverage. The authors in [8] use mobile BSs to provide downlink connectivity to ground users when their demand cannot be satisfied by the terrestrial station alone.

B. Massive MIMO

The main characteristic of massive MIMO is that it contains a massive number of antennas, generally 60 or more. It can take on several forms, but the main one used in this research is centralized massive MIMO where all the antennas are packed together in a single BS. To ensure that users receive their data stream with minimal interference, most forms of massive MIMO take advantage of spatial-division multiplexing. This form of multiplexing sends data streams at the same time and frequency [6]. What stops different users from receiving each other's signal is constructive and deconstructive interference with the antennas. The large number of antennas allows the BS to direct a signal at a specific user so only they will receive the signal. Massive MIMO BSs estimate channels using a pilot sequence that was sent by the user. One work that applies massive MIMO to mobile BSs is [11]. In this work, the authors applied a distributed algorithm price-based solution to UAVs to maximize the sum rate of all users and they did it by using convex relaxation to break the algorithm into three steps of access association, joint pilot assignment and power control and movement control. In this work, a massive MIMO array is attached to each drone and each drone controls a certain subset of users to control their pilot assignments and transmit powers with the goal of maximizing the sum of all the users' spectral efficiencies. [17] uses MIMO instead of massive MIMO and the only difference is the use of 2 antennas instead of 60 or more antennas.

C. Q-Learning

Q-Learning is an off-policy reinforcement algorithm that seeks to optimize the expected return based on Markov decision processes. The agent in Q-Learning starts by being in a certain state in the environment usually called the starting state and takes actions that will return a reward and transition the agent into a new state. In Q-Learning, an agent updates the q-value in its q-table where the rows are all the possible states, and the columns are all the possible actions. When an agent takes an action from an environment, it will update its q-value for the state it is in and the action that it took. All the values in the q-table are set to zero at the beginning and the table is updated using the following formula:

$$q_*^{new}(s,a) = (1 - \alpha)q(s,a) + \alpha\left(R_{t+1} + \gamma max\, q\left(s',a'\right)\right) \tag{1}$$

where α is the learning rate and γ is the discount rate.

The way the agent finds the best policy is by exploration and exploitation. When the agent is exploring the environment, it will select an action at random and when the agent is exploiting its environment, it will select the action with the highest q-value. At the beginning of training, the agent will explore its environment and will start to exploit it more and more as the episode number increases. This is an important step because the agent does not know anything about its environment in the beginning and once it starts to learn more, it will want to exploit it to find the best possible reward. One way this is achieved is by using the epsilon greedy strategy. This strategy sets a variable $\varepsilon = 1$, which decays exponentially and is updated using:

$$\alpha = \alpha_{end} + \left(\alpha_{start} - \alpha_{end}\right)\left(e^{-n_{step}\lambda}\right) \tag{2}$$

where α is the exploration rate, α_{end} is the ending exploration rate, α_{start} is the starting exploration rate, and λ is the exploration decay rate.

Reinforcement learning has been used in other UAV BS applications, such as determining the optimal positions for mobile BSs and [13] applies this to an emergency communication network. It may also be important to control the transmit power as well to minimize the interference between BSs, which is what [14] did by using a reinforcement learning algorithm to control transmit power and BS positioning to minimize user outage probabilities. Lastly, [17] uses reinforcement learning to analyze radio frequency channels to learn from past occupancy and conditions of the channels.

D. Deep Q-Learning

Unlike Q-Learning, which uses a q-table to keep track of its q-values, Deep Q-Learning uses a neural network where the input is the current state, and the output is the q-value for each action. The neural network can have any number of hidden layers and the main purpose of the neural network is to approximate the values of a q-table. For each action, state, reward and next state the agent experiences, this tuple is called experience replay. The experience replay includes the state of the environment, the action taken from that state, the reward given to the agent as a result of the previous state-action pair and the next state of the environment. Each experience replay is stored in an array called the replay memory up to some amount of experiences *N*. When the number of experiences gets larger than *N*, the first experience gets replaced with the most current experience. The replay memory gets sampled randomly to train the network, which breaks the correlation between successive samples to make the learning more efficient. When training the neural network, the loss is calculated by subtracting the q-value of the given state-action pair from the optimal q-value of the same state-action pair. The optimal q-value is calculated by passing the next state into the neural network to find the max q-value among all actions that can be taken in that state. The optimal q-values are not known at the beginning of training, so they are estimated using the neural network and get updated when the weights of the neural network get updated. To avoid instability, the optimal q-values are updated and calculated using a separate network called the target network. The target network is a copy of the original policy network, but only gets updated *x* timesteps.

Deep reinforcement learning has been used in [15] to find the best way to allocate resources in a distributed environment when the channel state information is not known. [16] uses this tool to also control transmit powers to mitigate interference, which helps maximize throughput.

E. Discussion of Related Work

Only one previous work [11] applies massive MIMO to UAV BSs, but they use a pricing algorithm to get within 90% the global optimum. None of the previous works have applied reinforcement learning or deep reinforcement learning to control the user association, pilot assignment and UAV movements to maximize the sum rate of the users. While [17] did use reinforcement learning, it does not take into account multiple BSs with multiple users. Therefore, this paper proposes a solution using Deep Q-learning, where each UAV is an agent that will try to maximize its reward based on the sum spectral efficiencies of all users. This approach is similar to [11] in that it is a distributed system and each UAV does not need to collect the full statistical Channel State Information (CSI), the locations of the other UAVs, the noise power and other network parameters. Also, since this is a distributed system, it does not encounter failure from a single point like centralized systems. Also, distributed systems scale better and have lower latencies than centralized systems.

III. Implementation Details

The deep reinforcement learning algorithm was written and simulated in python. Python was used so PyTorch and Gym can be used. Pytorch is used to create the target and policy neural networks in the Deep Q-Learning model and Gym is used to create the reinforcement learning environment. The Gym library is a toolkit for developing reinforcement learning algorithms and has many good environments to choose from when building the massive MIMO UAV agents and environment. A Gym environment class has four main functions that are needed to manage the agents in the environment. The *init* method is used to initialize all variables and constants, the *step* method executes one time step in the environment, the *reset* method resets the environment to the initial state, and the *render* method prints the current state of the environment to the screen.

A. Massive MIMO UAV BSs

All UAV BSs can move freely in a 500 m^2 × 500 m^2 area, but their heights are kept at a constant 100 m above the ground. The number of UAV BSs is set to {1, 2} and the number of users is set to {1, 2, 3, 4}. The users do not move through the environment and are randomly placed in the grid at the start of the simulation. The number of antennas in each UAV BS is set to {10, 20, 30, 40, 50, 100}. The range of transmit powers the UAV BSs can set for the ground users is in between [5, 500] mW and the total number of power levels is set to {3, 4, 5}. The path factor is set to 2, the average noise power is set to 10^{-8}, the length of each pilot sequence is set to 10 symbols and the total number of available pilot sequences is set to {2, 3, 4, 5, 6}.

B. Deep Reinforcement Learning Implementation

Since this problem is a MINLP problem, to find the best sum spectral efficiency, the pilot assignment is relaxed to allow a UAV to assign multiple pilot signals to a single user and the problem was broken down into two subproblems. In the first subproblem, the users are connected to the UAV that has the best SISO SNR. The next subproblem is the deep Q-Learning algorithm, which controls the UAVs' movements and the power allocation for each pilot sequence. For the implementation details of the reinforcement learning algorithm:

Agent: UAV BSs

State: Includes the position on the agents, the users the agents are connected to, the number of pilot sequences, and the number of power levels. To limit the total number of states, the agents operate in a square grid, and the total number of transmit powers are divided into *n* power levels. The level of different transmit powers is divided evenly between the max and min user transmit powers.

Action: Includes moving up, down, left, right, increasing user transmit power assigned to a pilot sequence, and decreasing user transmit power assigned to a pilot sequence.

Reward: Based on the sum of the spectral efficiency for the users connected to UAVs. The spectral efficiency is calculated by dividing the capacity from (3) by *B* Hz.

$$C_g = B\log_2(1+\gamma_g) \tag{3}$$

The capacity is then calculated with (3), which includes channel-estimation error, the type of linear spatial multiplexing/demultiplexing, power control, and noncoherent inter-cell interference (4) [11].

Lastly, to get the reward from the spectral efficiency equation, it is multiplied by the power level the UAV chooses for a pilot sequence divided by the max power level. This was done to encourage the agents to choose only one pilot sequence.

$$R_g = C_g\left(\frac{p_{gw}}{p_{max}}\right) \tag{5}$$

The neural network in this article has one hidden layer with 500 nodes. The input to the neural network is one-hot encoded, which makes the input have as many nodes as the number of states, and the number of nodes at the output is determined by the total possible actions that the agent can take. After each action, the agent stores the action-reward pair in memory. After the agent takes a certain number of actions, it updates the target network based on a random batch of action-reward pairs from memory using the Adam optimizer. The purpose of the target network is to help reduce instability when both the training q-values and the optimum q-values are both being updated throughout the simulation. The loss is calculated using mean square error between the q-value calculated in the training network and the q-value calculated in the target network. The agent repeats the above process until the training is over.

C. Relaxed Centralized Solution

The simulations were compared to a relaxed centralized solution to see how close they were to the max possible reward. This problem is solved by maximizing (6), which finds the x,y, and z positions of the UAVs. This returns the largest sum spectral efficiencies across all users. This relaxed centralized solution is not the main focus of the paper because the UAVs do not know the locations of the users. Also, a centralized system is more prone to failure if a single component fails while a decentralized system is more resilient to failure.

$$Maximize\colon \sum_{g\in G} C_g(x,y,z) \tag{6}$$

In (6), all interferences are ignored and all users are assumed to be using separate pilot sequences. Also, $C_g(x,y,z) = Blog_2(1+\gamma_g(x,y,z))$, and since $\gamma_g(x,y,z)$ >> 1, $C_g(x,y,z)$ can be approximated.

$$\approx Blog_2(\gamma_g(x,y,z))$$

$$\leq Blog_2(M\tau\rho_g p_0 \zeta_{gg}^2 H_{gg}^2(x,y,z))$$

$$= Blog_2\left(\frac{M\tau\rho_g p_0 \zeta_{gg}^2}{d_{gg}^x(x,y,z)}\right)$$

$$= Blog_2(M\tau\rho_g p_0 \zeta_{gg}^2) - xBlog_2(d_{gg}(x,y,z)) \tag{7}$$

When looking at (7), the first term is constant since none of the variables are a function of x, y, or z. Another way to write this maximization problem is to write it as a minimization of the second term.

$$Minimize\colon \sum_{g\in G} -log_2(d_{gg}(x,y,z)) \tag{8}$$

The exact solution to this problem is approximated by calculating a 20000 x 20000 grid of all the possible x and y positions of the UAV. This finds a solution very close to the

$$\gamma_{gw}(\tilde{\mathbf{p}}) = \frac{(M-|\mathcal{G}_{a(g)}|)\tau\rho_g\beta_{gg}^2 p_{gw}}{(1+\tau\mathcal{E})(1+\sum_{g'\in\mathcal{G}}\sum_{w'\in\mathcal{W}}\mu_{g'g}(\boldsymbol{\mu})p_{g'w'}) + (M-|\mathcal{G}_{a(g)}|)\sum_{g'\in\mathcal{I}_g\setminus g}\sum_{w'\in\mathcal{W}}\rho_{g'}\beta_{g'g}^2 p_{g'w'}} \tag{4}$$

optimum solution because there is only 25 mm separation for each square in the grid.

IV. Simulation Results

For the experiment, the data that was recorded was the cumulative reward and it is the total reward the agent received during each episode. This benchmark is used because it shows the agent gradually choosing better actions that return a higher reward more often in a single episode. The simulation parameters are stored in Table 2 and Table 1 describes the meaning of all the parameters in Table 2. In Figure 1, there are 2 UAVs, 2 users, and 2 pilot sequences to choose from. In the beginning of the graph, there are four different colors stacked on top of each other. The blue line on the bottom represents one or both of the UAVs assigning the first pilot sequence to the first user. The orange line above the blue line represents one or both of the UAVs assigning the second pilot sequence to the first user. The green line above the orange line represents one or both of the UAVs assigning the first pilot sequence to the second user. Lastly, the red line above the green line represents one or both of the UAVs assigning the second pilot sequence to the second user. Since this is the cumulative reward, for each episode there are 2000 iterations where the UAV can change the pilot sequence assigned to a user. In the beginning of the simulation, the UAVs do not know which pilot sequence to assign to the users, so it picks randomly. As the simulation progresses, the UAVs learn that if a user is assigned a pilot sequence by one or the other UAV, they or the other UAV should assign the other pilot sequence to the other user. This can be seen near the end of the simulation where there is only a green line above an orange line, or a red line above a blue line (Figure 1). This means that either the first user was assigned the first pilot sequence and the second user was assigned the second pilot sequence or the first user was assigned the second pilot sequence and the second user was assigned the first pilot sequence. There is also an average in the graph which is represented by the black line on top. The average takes into account the past 50 episodes and can be seen increasing until it levels out at about the 20000th episode.

The optimum cumulative reward is found by finding the optimum location for the UAVs using the relaxed centralized solution and then multiplying it by the number of iterations for one episode in the simulation. The optimum cumulative reward is shown in the graph by a yellow horizontal line on the top of the graph and it is also labeled for clarity. The simulation in Figure 1 can be seen to come very close to the optimum cumulative reward found using the relaxed centralized solution. In this simulation, the drone network average was able to come within 95% of the cumulative relaxed centralized solution.

A. Discussion

The UAVs were able to find the optimum solution even though the reward function was not fully convex. The function that the reinforcement learning algorithm is optimizing over greatly determines how quickly a solution can be found. Further investigation will look into adding more UAVs, more users, and simulations where there are more users than pilot sequences. Further investigation will also look into different ways the policy network can be updated to improve convergence when more agents and users are added. This will be done to test results with more realistic scenarios where there are more users and not enough pilot sequences for each user.

V. Conclusion

In this article, deep reinforcement learning was implemented and evaluated to optimize the sum spectral efficiency of ground users. To verify this, a simulation was built in Python using OpenAI's Gym library to create a custom agent, reward function, and environment. Details are included on how the deep reinforcement learning algorithm is set up and the convex relation techniques used to help the aerial drone find the maximum spectral efficiency. Lastly, the results are compared to the log of the euclidean distance to see how the simulations compare to the near optimum sum spectral efficiency.

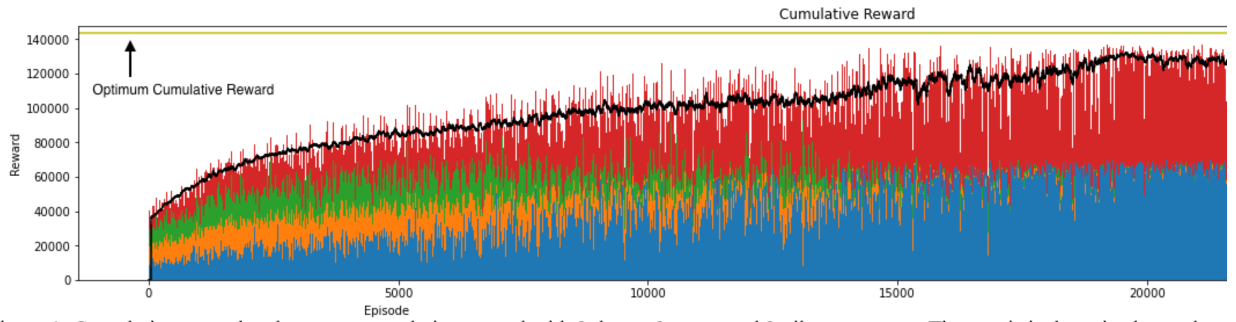

Figure 1: Cumulative reward and average cumulative reward with 2 drone, 2 users, and 2 pilot sequences. The x-axis is the episode number and the y-axis in the cumulative reward.

References

[1] R. Iyer and E. Ozer, "Visual IoT: Architectural Challenges and Opportunities; Toward a Self-Learning and Energy-Neutral IoT," in IEEE Micro, vol. 36, no. 6, pp. 45-49, Nov.-Dec. 2016, doi: 10.1109/MM.2016.96.

[2] V. J. Hodge, S. O'Keefe, M. Weeks and A. Moulds, "Wireless Sensor Networks for Condition Monitoring in the Railway Industry: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 3, pp. 1088-1106, June 2015, doi: 10.1109/TITS.2014.2366512.

[3] E. Björnson, E. G. Larsson and T. L. Marzetta, "Massive MIMO: ten myths and one critical question," in IEEE Communications Magazine, vol. 54, no. 2, pp. 114-123, February 2016, doi: 10.1109/MCOM.2016.7402270.

[4] M. Mozaffari, W. Saad, M. Bennis, Y. -H. Nam and M. Debbah, "A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems," in IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2334-2360, thirdquarter 2019, doi: 10.1109/COMST.2019.2902862.

[5] M. N. Boukoberine, Z. Zhou and M. Benbouzid, "Power Supply Architectures for Drones - A Review," IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society, 2019, pp. 5826-5831, doi: 10.1109/IECON.2019.8927702.

[6] T. L. Marzetta, "Massive MIMO: An Introduction," in Bell Labs Technical Journal, vol. 20, pp. 11-22, 2015, doi: 10.15325/BLTJ.2015.2407793.

[7] A. Vallamreddy and P. Indumathi, "Outage performance analysis of MIMO cognitive radio network users in fading environment," 2014 2nd International Conference on Devices, Circuits and Systems (ICDCS), 2014, pp. 1-4, doi: 10.1109/ICDCSyst.2014.6926207.

[8] W. Shi et al., "Multiple Drone-Cell Deployment Analyses and Optimization in Drone Assisted Radio Access Networks," in IEEE Access, vol. 6, pp. 12518-12529, 2018, doi: 10.1109/ACCESS.2018.2803788.

[9] A. Fotouhi, M. Ding and M. Hassan, "Flying Drone Base Stations for Macro Hotspots," in IEEE Access, vol. 6, pp. 19530-19539, 2018, doi: 10.1109/ACCESS.2018.2817799.

[10] A. Fotouhi, "Towards intelligent flying base stations in future wireless network," 2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2017, pp. 1-3, doi: 10.1109/WoWMoM.2017.7974302.

[11] Z. Guan, N. Cen, T. Melodia and S. M. Pudlewski, "Distributed Joint Power, Association and Flight Control for Massive-MIMO Self-Organizing Flying Drones," in IEEE/ACM Transactions on Networking, vol. 28, no. 4, pp. 1491-1505, Aug. 2020, doi: 10.1109/TNET.2020.2985972.

[12] M. F. Ahmed, E. M. Naveed, P. Nar and S. K. Jindal, "Design of an Autonomous drone for IoT deployment analysis," 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), 2019, pp. 1-4, doi: 10.1109/ViTECoN.2019.8899609.

[13] P.V. Klaine, J.P.B Nadas, R.D. Souza, M.A. Imran, “Distributed Drone Base Station Positioning for Emergency Cellular Networks Using Reinforcement Learning”. *Cogn Comput* 10, 790–804, 2018. pp. 790-804, doi: 10.1007/s12559-018-9559-8

[14] R. de Paula Parisotto, P. V. Klaine, J. P. B. Nadas, R. D. Souza, G. Brante and M. A. Imran, "Drone Base Station Positioning and Power Allocation using Reinforcement Learning," 2019 16th International Symposium on Wireless Communication Systems (ISWCS), 2019, pp. 213-217, doi: 10.1109/ISWCS.2019.8877247

[15] J. Jang, H. J. Yang and S. Kim, "Learning-Based Distributed Resource Allocation in Asynchronous Multicell Networks," 2018 International Conference on Information and Communication Technology Convergence (ICTC), 2018, pp. 910-913, doi: 10.1109/ICTC.2018.8539654.

[16] G. Zhao, Y. Li, C. Xu, Z. Han, Y. Xing and S. Yu, "Joint Power Control and Channel Allocation for Interference Mitigation Based on Reinforcement Learning," in IEEE Access, vol. 7, pp. 177254-177265, 2019, doi: 10.1109/ACCESS.2019.2937438.

[17] L. Bondan, M. A. Marotta, L. R. Faganello, J. Rochol and L. Z. Granville, "ChiMaS: A spectrum sensing-based channels classification system for cognitive radio networks," 2016 IEEE Wireless Communications and Networking Conference, 2016, pp. 1-7, doi: 10.1109/WCNC.2016.7564911.

TABLE I.

Variables	*Meaning*
Batch Size	Number of samples used to train the neural network
Gamma	Discount rate
Starting Epsilon	Staring value of epsilon decay
Ending Epsilon	Ending value of epsilon decay
Epsilon Decay	Value to decrease epsilon by
Target Update	Updates target network every *N* Episodes
Memory Size	Max number of experiences in replay memory
Learning Rate	Learning rate of neural network
Number of Episodes	Max number of episodes in simulation
Max Steps Per Episode	Max number of steps before episode ends

TABLE II.

Values of Variables used in Simulation	
Variables	*Values*
Batch Size	10
Gamma	0.99
Starting Epsilon	0.9
Ending Epsilon	0.001
Epsilon Decay	$4*10^{-8}$
Target Update	60
Memory Size	1000
Learning Rate	0.0001
Number of Episodes	50000
Max Steps Per Episode	2000