Optimal Trajectory Learning for UAV-Mounted Mobile Base Stations using RL and Greedy Algorithms

Adhitya Bantwal Bhandarkar
Communication and Information Sciences Laboratory (CISL)
Dept. of Electrical and Computer Engineering
University of New Mexico
Albuquerque, USA
abhandarkar@unm.edu

Sudharman K Jayaweera
Communication and Information Sciences Laboratory (CISL)
Dept. of Electrical and Computer Engineering
University of New Mexico
Albuquerque, USA
jayaweera@unm.edu

***Abstract*—This paper designs Artificial Intelligence (AI) method, to determine an optimal trajectory for an Unmanned Aerial Vehicle (UAV) mounted mobile base station to maximize its coverage of distinct users. Determining such an optimal trajectory for arbitrarily distributed users over an area is, in general, difficult and there is no closed-form solution. Since the users are arbitrarily located the method must adapt accordingly. To accomplish this, the problem is formulated in a way that is compatible with Reinforcement Learning (RL) and two AI approaches are designed to learn an optimal trajectory. The first uses Deep Reinforcement Learning (DRL) implemented with a Deep Q-Network (DQN) while the second is a reward-based greedy algorithm. It is shown that these new algorithms significantly outperform the state-of-the-art previously proposed deep learning based approaches. Moreover, the simple AI-based greedy approach is shown to perform close to the DQN-aided DRL algorithm at a much lower computational complexity at least in the type of scenarios considered in this paper.**

***Index Terms*—Deep Q-Network (DQN), Deep Reinforcement Learning (DRL), optimal trajectory learning, Unmanned Aerial Vehicles (UAVs), wireless user coverage, greedy algorithm.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) have gained popularity over the years for use in military and civil applications [1], [2]. In civilian areas, UAVs have gained traction for use in wireless communication systems. In particular, they have been considered as flying base stations to provide wireless coverage to areas that are inaccessible via traditional mobile base stations [3], [4]. For example, wireless transceivers mounted on UAVs can act as portable mobile base stations that can be deployed in scenarios including floods, earthquakes, wars and other similar disasters [5].

Similar to a traditional fixed base station, a UAV-mounted base station will have a limited radius of coverage. However, unlike a traditional fixed base station, a UAV-mounted base station's flying and serving time will be limited by its battery life. However, a UAV-mounted base station allows for choosing a path that maximizes the number of distinct ground users covered [6]. The total flying time of the UAV can be divided into intervals of equal duration. At each time instant, the UAV flies over a location covering a set of users and for the next time interval it moves to another location covering a different set of users [6]. The set of locations the UAV traverses from the beginning to exhaustion of its energy is called its trajectory [5], [6]. This gives rise to the optimal trajectory planning problem for a UAV-mounted base station where optimality can be defined as maximizing the number of distinct users served by the UAV before it runs out of battery life.

The trajectory has to be fair in a sense that no set of users must be favored over the other. In other words the UAV should provide a fair coverage. Choosing such a trajectory that covers as many users as possible can be difficult. Finding an optimal trajectory using brute force method can be computationally expensive and inefficient. In this scenario, using artificial intelligence (AI) and learning based algorithms can be a promising approach.

Over the past few years, several innovative techniques have been proposed to provide better coverage to ground users by deploying UAVs. For example, the authors of [5] discuss how UAVs can be placed in such a way that every ground terminal is connected to at least one base station. This ensures that every user is connected to a mobile base station. When users are spread across a large geographical area, however, a large number of UAVs are required which may not always be feasible. Providing coverage for arbitrarily distributed set of users by strategically placing a single UAV is discussed in [7]. The authors developed a modified knapsack algorithm to find the optimal height and coordinates for the UAV mounted base station to cover as many users with guaranteed data rates. However, a single stationary UAV might not always be able to serve all users owing to its limited radius of coverage. The authors of [8] discussed deployment of multiple UAV base stations in disaster struck areas. The authors used

978-1-6654-2854-5/21/$31.00 ©2021 IEEE

Artificial Bee Colony (ABC) algorithm to place the UAVs in a way that maximizes network throughput. However, the energy limitations of UAVs were not taken into account. The authors of [9] discussed deployment of a UAV in a dynamic environment where the UAV has to provide coverage to a non-stationary set of users. The authors used majority vote rule wherein the UAV moves to a spatial location with the greatest number of users. This, however, has some limitations: the majority rule will be biased against smaller sets of users and such users might not be provided a fair coverage. Using UAV as a relay between a fixed base station and a mobile terminal to reduce the probability of outage was considered in [10]. This method indeed reduces the outage for users located at places that are outside the coverage of the base station. However, if there are clusters of users, each of which are outside the coverage area and are at different places, then additional relays would need to be deployed to provide coverage to each such cluster of users which might not be possible or desirable.

For arbitrarily distributed users, [11] used a genetic algorithm based K-means (GAK) to partition the area into cells and then used Q-Learning algorithm for initial placement and movement of multiple UAVs based on Quality of Experience (QoE). The authors of [6] extended the approach of [11] by using Q-learning and Deep Q-Learning (DQL) based approaches for finding a path that maximizes fairness and number of users covered using single and multiple UAVs. To ensure all users are treated fairly, [6] incorporates Jain's Fairness index [6] into the reward function of Q-learning and DQL algorithms. Using this modified reward function, [6] was shown to outperform the approaches proposed in [11].

Although the Q-Learning and DQL based methods of [6] seem to outperform previous results of [11], a closer look shows that it has some serious drawbacks. For instance, the authors used coverage history of all the users up to that time instant, along with location of the UAV and its battery level, as input to the Q-Learning and DQL algorithms. As a result, the dimension of the input changes as the number of users is changed. Hence, the number of users must be known in advance and the resulting ML model can only be used when the number of users is exactly equal to the number of users for which the DQL agent was trained.

In this paper, we first recast the optimal trajectory planning problem considered in [11] and [6] by generalizing it as a problem of maximizing the coverage for distinct users. Then, we propose two AI-based approaches to find an optimal trajectory for a UAV-mounted base station that not only avoids the drawbacks of [6] but also significantly outperform both [6] and [11].

The remainder of the paper is organized as follows: Section II formulates the optimal trajectory planning problem for a UAV-mounted base station. Section III discusses the proposed methods for learning an optimal trajectory for a UAV. Section IV presents performance results of the proposed methods and compares them to the state-of-the-art methods found in the existing literature. Finally, section V concludes the paper by discussing further improvements that can be pursued in future.

II. Problem Formulation

A. System Model

As in [6], a square geographical area of length L is divided into M number of square cells each of identical size as shown in Fig. 1. An arbitrary, and unknown, number of users are assumed to be spread uniformly across this area. We assume that the UAV maintains a fixed altitude precluding the actions of moving up or down. In many situations this may make sense in order to provide uniform coverage while simplifying interference management among multiple UAVs and subscribers [6], [12]. For simplicity, the UAV is restricted to hover only in the corners and centers of the cells, so that there are $2\sqrt{M}(\sqrt{M}+1)+1$ possible hovering points. If we assume that the coverage radius of the UAV base station is $l/2$, where l is the side-length of a square cell, then the UAV base station will be able to provide coverage to any user from one of the possible hovering points.

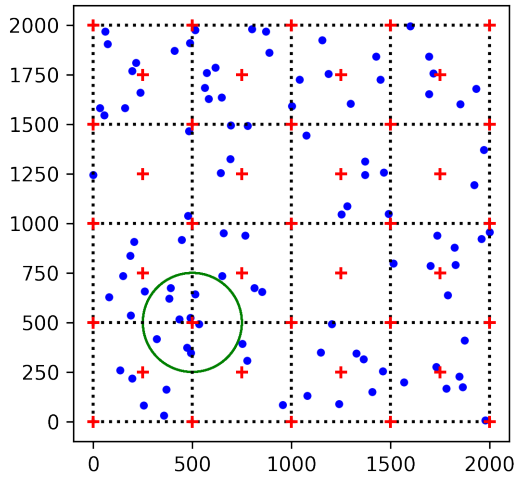

Fig. 1. A UAV hovering at location (500,500). The UAV coverage radius is represented by the green circle. The blue dots indicate the users and the 41 red crosses represent the possible hovering points.

The limited battery life of a UAV, however, imposes a constraint on the total flying time of the UAV. As with [6], [11], we divide the total flying time of the UAV into N slots of equal length. At each time slot, the UAV can be in any of the $2\sqrt{M}(\sqrt{M}+1)+1$ hovering points providing service to users located within the coverage radius. As can be seen from Fig. 1, the possibility of all ground users being covered when UAV is hovering over any fixed point is highly unlikely justifying the need for a UAV to traverse a trajectory over the given geographical area to find and provide coverage to distributed users.

Although a UAV is capable of moving in any direction, for the sake of simplicity, we assume that the UAV is restricted to move in nine directions namely: North, North-East, East, South-East, South, South-West, West, North-West or be stationary. Note that, despite this restriction, a UAV will still be able to provide coverage to any location starting from an arbitrary hovering point in finite number of steps.

B. Deep Q-Learning

A reinforcement learning algorithm consists of an environment and an agent. At each time step t, the agent observes the state of the environment at that time represented as S_t, takes an action appropriate for that state and time given by a_t and receives a scalar reward r_t depending on the action and the environment's transition to a new state S_{t+1}. For any given state, the agent should take an action a that maximizes the sum of discounted expected future rewards [13]. In the case of Q-Learning, this is done by looking up the Q-table and selecting an action that has the highest expected future rewards:

$$a_t = \underset{a}{\operatorname{argmax}}\ Q(S_t, a)$$

Note that, $Q(S_t, a)$ is a function of the two variables S_t and a. A Q-table approach to learning the function $Q(S_t, a)$ works if both S_t and a were to take values on finite sets. Even then, unless the number of actions and possible states are relatively small, the Q-table approach may not be convenient in practice.

The Deep Q-Network (DQN) approach for RL proposed in [15], instead uses a deep neural network (DNN) as a function estimator to learn the $Q(S_t, a)$. The DQN makes use of an experience replay [15] where a replay memory is used to store tuples of transitions consisting of: The State S_t, the action a_t taken for that state, reward obtained by taking that action r_t and the resulting new state S_{t+1}. At every time step, the DQN is trained by sampling a random batch of tuples from this experience replay memory. The Algorithm 1 details the DQN-based RL approach.

Algorithm 1 DQN Algorithm

Initialize policy network $Q(s, a)$ with random weights
Set target network $Q'(s, a)$ with the same weights
Set $\varepsilon = 1$
Create a replay memory with
for episode = 1 to M **do**
  for time t = 1 to N **do**
    Generate a random number w between 0 and 1
    if $\varepsilon > w$ **then**
      Take a random action a_t
    else
      Take action: $a_t = \operatorname{argmax}_a Q(S_t, a)$
    Store transition: $(S_t\ ,\ a_t\ ,\ r_t\ ,\ S_{t+1})$ into buffer
    Sample random batch of transitions from the buffer
    if Episode ends at $t + 1$ **then**
      Set $y_t = r_t$
    else
      Set $y_t = r_t + \gamma \max_{a'} Q'(S_{t+1}, a')$
    Fit the policy network on the data: (S_t, y_t).
    Reduce the value of ε
    Every K steps set weights of target equal to policy network.

C. Method proposed in [6]

Authors of [6] formulated the trajectory finding problem for a UAV as a reinforcement learning problem and used the Q-learning algorithm to solve it. The state used in [6] is a vector consisting of coverage of each ground user up to that instant. Thus the state is of size $(N + 3) \times 1$ where N represents the number of users and the three additional elements represent the X and Y coordinates of the current location of the UAV and the energy remaining in the UAV, as shown below:

$$S_t = \left[\sum_{i=0}^{t} Cov_1^i, \sum_{i=0}^{t} Cov_2^i, \cdots, \sum_{i=0}^{t} Cov_N^i, x_t, y_t, e_t\right]$$

where $Cov_n^t \in \{0, 1\}$, for $n \in \{1, \cdots, N\}$, denotes whether the n-th ground user has the coverage at time instant t or not:

$$Cov_n^t = \begin{cases} 1, & \text{if the user is inside coverage area of UAV} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The authors defined the reward function for the t-th time instance as:

$$r_t = p_t + \begin{cases} \sum_{n=1}^{N} Cov_n^t \times (F_t - F_{t-1}), & \text{if } F_{t-1} \neq 1. \\ \sum_{n=1}^{N} Cov_n^t \times F_t, & \text{otherwise.} \end{cases} \tag{2}$$

where F_t is the Jain's fairness [15] at time step t for N ground users, defined as:

$$F_t = \frac{(\sum_{n=1}^{N} \sum_{i=0}^{t} Cov_n^i)^2}{N(\sum_{n=1}^{N} (\sum_{i=0}^{t} Cov_n^i)^2)}$$

and p_t is a penalty function which is non-zero only if the UAV attempts an action that will result in the UAV flying out of the designated area:

$$p_t = \begin{cases} 0, & \text{if } 0 \leq\ x_{t+1}, y_{t+1} \leq\ L \\ \text{P}, & \text{otherwise.} \end{cases} \tag{3}$$

In (3), (x_{t+1}, y_{t+1}) denotes the resulting coordinates of the UAV if action a_t is taken at the t-th time step and $P < 0$ is the penalty for selecting an action that will result in flying outside the desired coverage area. A large magnitude for the penalty P discourages the UAV from taking actions in future that would result in UAV flying out of the desired area. Note that, if the number of users covered at that instant were taken as the reward function directly, then there is a possibility that the DQN-UAV agent may get stuck at a hovering point and not move further because any movement may result in a reduced reward. The reward function (2) avoids this because if the UAV hovers at the same location twice in a row then the fairness for that time instant will be less than that of previous time instant, making the reward for that time instant negative.

III. Proposed Method

A. Proposed DQN method

Note that, the dimension of the state used in [6] will change if the number of users are changed. Since the state is the input to the DQN, the neural network will have to be modified each

time the number of users are varied. Hence, in practice it is important that the dimensions of the state be independent of the number of users. We propose a new DQN based approach in which the state is defined to be an array whose dimensions are determined only by the number of hovering points.

In this approach, the state is made of the number of users covered at each hovering point up to that time instant along with the energy remaining in the UAV and its coordinates. Thus, the state will have $2\sqrt{M}(\sqrt{M}+1)+4$ elements unlike [6] which had $N+3$ elements. This eliminates the dependency of the state on number of users and ensures that the dimension of the state remains fixed regardless of the number of users. Fig. 2 shows the state in the proposed method which has been reshaped into a matrix. Reshaping the state as a matrix allows to use Convolutional Neural Network (CNN) in the DQN which is better at capturing spatial features.

This approach indeed was able to find an optimal trajectory at a significantly lower number of training iterations compared to [6] as discussed in the next section. The state in Fig. 2 can further be modified by augmenting information from previous time steps to create a tensor as shown in Fig. 3. In this case, the number of previous time steps included in the state tensor is called the depth. Increasing depth resulted in the DQN converging a little faster and with improved consistency but only when using a larger learning rate.

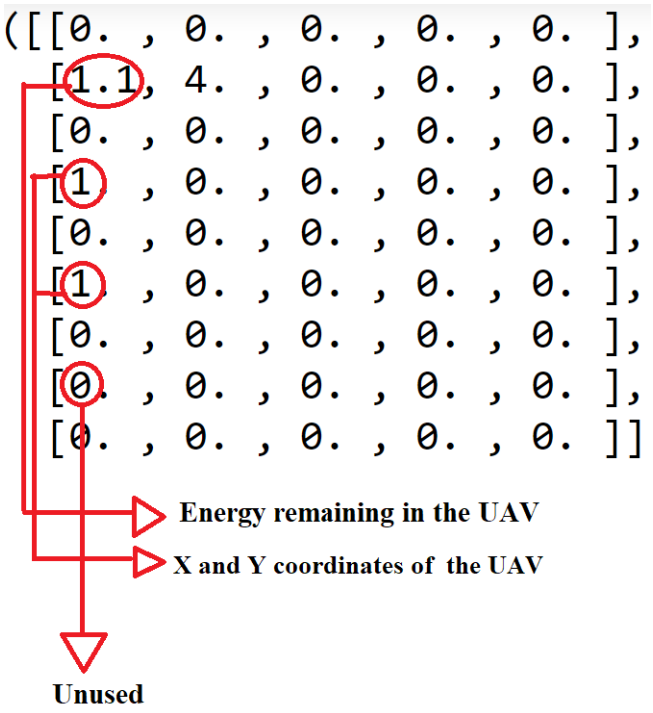

Fig. 2. A sample state for depth = 1

B. Proposed AI based greedy algorithm

For a time instant t and corresponding state S_t, the proposed greedy algorithm chooses an action a_t that yields immediate maximum reward r_t. The greedy algorithm does not take into account whether the selected action will have unfavourable results in the long run. Although the algorithm is relatively simple compared to the DQN based approach discussed in the previous section, it is nevertheless effective. Indeed, as we will discuss in the next Section, it was able to perform almost as good as the DQN method with significantly lower computational complexity and outperform all previously proposed methods in the literature.

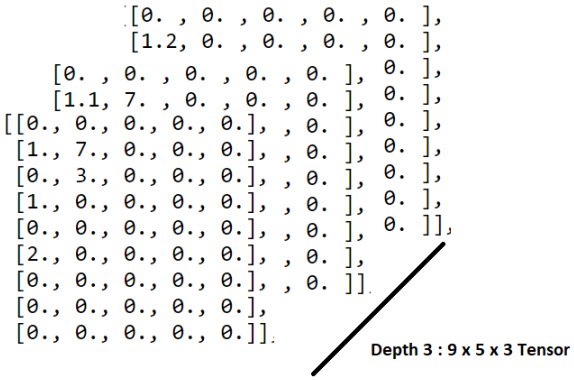

Fig. 3. A sample state for depth = 3

IV. Simulation Results

As in [6] we consider an area of $2000m \times 2000m$ divided into $M = 16$ square cells of same size resulting in 41 hovering points. For the simulation, 60 to 100 users were uniformly distributed over this area.

As shown in Fig. 2, the state in this case is a 9×5 matrix, 41 elements of which represent the number of users covered up to that time at 41 different hovering points, three more elements record the UAV's X and Y coordinates and its remaining energy. Remaining unused elements are set to 0. The parameters used in the simulation are summarized in Table I and II.

TABLE I Simulation parameters

Area	2000m x 2000m
Number of Blocks (B)	16
Size of Blocks	500m x 500m
Number of Hovering points	41
Number of users	60 to 100
Number of time slots	12
Radius of coverage of UAV	250m
Penalty (P)	-10

TABLE II Specifications of the Convolutional Neural Network

Input Shape	9x5x1
Kernel Size	2 x 2
Padding	Same
Layers	3
Filters in each layer	40, 45 , 55
Activation function	Sigmoid
Optimizer	Adam
Loss Function	Huber
Gamma	0.9
Epsilon decay rate	0.001
Learning rate (lr)	0.001, 0.0001
Batch Size	$0.4 \times \min(timesteps, 2400)$

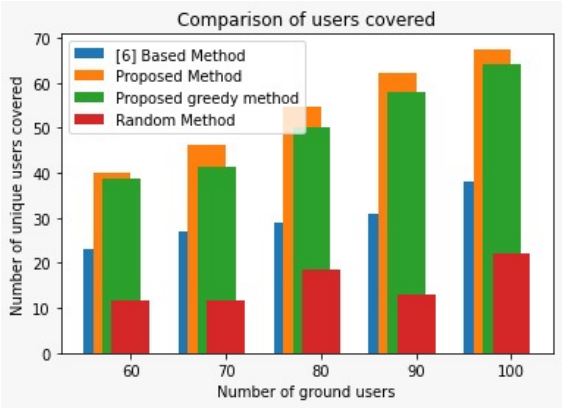

Fig. 4. Number of users covered by the DQN-UAV agent for different number of ground users

Figure 4 shows the number of distinct users covered with the two proposed methods of this paper, previously proposed method in [6] and random action selection policy for 60 to 100 ground users. There is a significant difference in the number of unique users covered by [6] and the proposed methods. From Fig. 4 it can be inferred that roughly twice as many users are covered using the proposed method in contrast to the method based on [6]. Figure 4 also shows that the proposed greedy method is able to perform almost as good as the proposed DQN based method and is significantly better than [6]. Note that, there is no training requirement for the proposed greedy method as for the DQN based proposed method and method of [6]. This is because at each time step, the greedy algorithm simply selects the action that maximizes the instantaneous reward which is computationally inexpensive.

Figures 5 and 6 show the number of training epochs that are needed to find an optimal trajectory at learning rates of 10^{-4} and 10^{-3}, respectively. From Fig. 5, we can observe that the DQN is able to find an optimal trajectory in roughly 500 training iterations when learning rate is 10^{-4}. Figure 6 shows that the performance with the larger learning rate is inferior which can be attributed to DQN weights not being able to converge due to oscillations. Figures 7 and 8 show

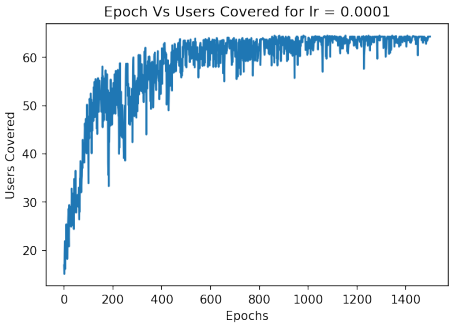

Fig. 5. Number of users covered by the DQN-UAV as training progresses at lr = 10^{-4}

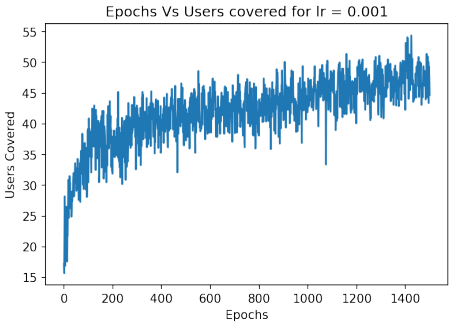

Fig. 6. Number of users covered by the DQN-UAV as training progresses at lr = 10^{-3}

the impact of depth parameter in the state on the number of training iterations required in finding an optimal trajectory averaged over ten random distribution of users at learning rates of 10^{-4} and 10^{-3}, respectively. It appears that higher depths are advantageous only when learning rate is high. Figures 9 and 10 show the variance in number of users covered after each epoch for learning rate 10^{-4} and 10^{-3}. It can be observed that the variance is less when the smaller learning rate is used signifying that the DQN performs better when lower learning rates are used.

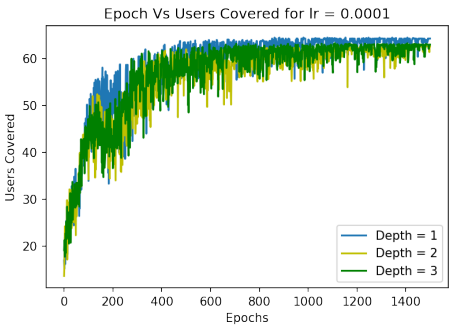

Fig. 7. Average users covered at each depth for 100 at learning rate of 10^{-4}

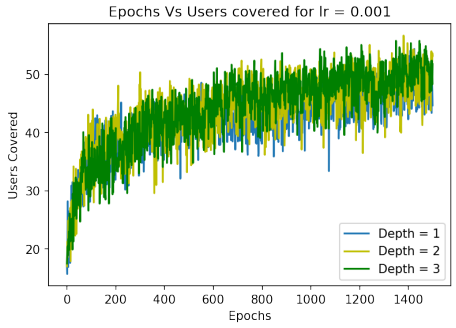

Fig. 8. Average users covered at each depth at learning rate of 10^{-3}

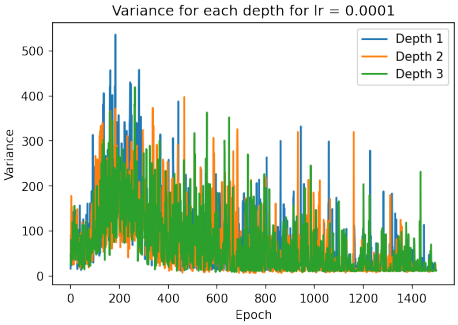

Fig. 9. Variance in number of users covered at each depth for 100 users

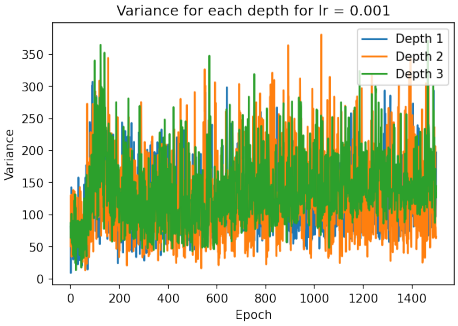

Fig. 10. Variance in number of users covered at each depth for 100 users

V. Conclusion

We developed two AI based algorithms for finding an optimal trajectory for a UAV-mounted base station that provides coverage to maximum number of distinct ground users. The performance of the two approaches were compared with existing state-of-the-art approach. The results show that the trajectories obtained using the proposed methods were able to cover significantly more number of users than previous approaches. In addition, the proposed greedy algorithm was shown to perform very close to the proposed DQN-aided DRL algorithm but at a much lower computational complexity.

References

[1] M. Böyük, R. Duvar and O. Urhan, "Deep Learning Based Vehicle Detection with Images Taken from Unmanned Air Vehicle," 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), 2020, pp. 1-4, doi: 10.1109/ASYU50717.2020.9259868.

[2] S. Hayat, E. Yanmaz and R. Muzaffar, "Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint," in IEEE Communications Surveys and Tutorials, vol. 18, no. 4, pp. 2624-2661, Fourthquarter 2016, doi: 10.1109/COMST.2016.2560343.

[3] Y. Zeng, R. Zhang and T. J. Lim, "Wireless communications with unmanned aerial vehicles: opportunities and challenges," in IEEE Communications Magazine, vol. 54, no. 5, pp. 36-42, May 2016, doi: 10.1109/MCOM.2016.7470933.

[4] R. I. Bor-Yaliniz, A. El-Keyi and H. Yanikomeroglu, "Efficient 3-D placement of an aerial base station in next generation cellular networks," 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1-5, doi: 10.1109/ICC.2016.7510820.

[5] J. Lyu, Y. Zeng, R. Zhang and T. J. Lim, "Placement Optimization of UAV-Mounted Mobile Base Stations," in IEEE Communications Letters, vol. 21, no. 3, pp. 604-607, March 2017, doi: 10.1109/LCOMM.2016.2633248.

[6] H. V. Abeywickrama, Y. He, E. Dutkiewicz, B. A. Jayawickrama and M. Mueck, "A Reinforcement Learning Approach for Fair User Coverage Using UAV Mounted Base Stations Under Energy Constraints," in IEEE Open Journal of Vehicular Technology, vol. 1, pp. 67-81, 2020, doi: 10.1109/OJVT.2020.2971594.

[7] C. Lai, C. Chen and L. Wang, "On-Demand Density-Aware UAV Base Station 3D Placement for Arbitrarily Distributed Users With Guaranteed Data Rates," in IEEE Wireless Communications Letters, vol. 8, no. 3, pp. 913-916, June 2019, doi: 10.1109/LWC.2019.2899599.

[8] J. Li, D. Lu, G. Zhang, J. Tian and Y. Pang, "Post-Disaster Unmanned Aerial Vehicle Base Station Deployment Method Based on Artificial Bee Colony Algorithm," in IEEE Access, vol. 7, pp. 168327-168336, 2019, doi: 10.1109/ACCESS.2019.2954332.

[9] Z. Wang, L. Duan and R. Zhang, "Adaptive Deployment for UAV-Aided Communication Networks," in IEEE Transactions on Wireless Communications, vol. 18, no. 9, pp. 4531-4543, Sept. 2019, doi: 10.1109/TWC.2019.2926279.

[10] S. Zhang, H. Zhang, Q. He, K. Bian and L. Song, "Joint Trajectory and Power Optimization for UAV Relay Networks," in IEEE Communications Letters, vol. 22, no. 1, pp. 161-164, Jan. 2018, doi: 10.1109/LCOMM.2017.2763135.

[11] X. Liu, Y. Liu and Y. Chen, "Reinforcement Learning in Multiple-UAV Networks: Deployment and Movement Design," in IEEE Transactions on Vehicular Technology, vol. 68, no. 8, pp. 8036-8049, Aug. 2019, doi: 10.1109/TVT.2019.2922849.

[12] Ding, Ming & Lopez-Perez, David. (2016). Please Lower Small Cell Antenna Heights in 5G 10.1109/GLOCOM.2016.7842150.

[13] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 2018.

[14] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[15] Jain, R.; Chiu, D.M.; Hawe, W. (1984). "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems" DEC Research Report TR-301.