User Coverage Maximization for a UAV-mounted Base Station Using Reinforcement Learning and Greedy Methods

Adhitya Bantwal Bhandarkar
Comm. & Info. Sciences Lab (CISL)
ECE Department,
University of New Mexico,
Albuquerque, USA
abhandarkar@unm.edu

Sudharman K. Jayaweera
Comm. & Info. Sciences Lab (CISL)
ECE Department,
University of New Mexico,
Albuquerque, USA
jayaweera@unm.edu

Steven A. Lane
Air Force Research Laboratory (AFRL)
Space Vehicles Directorate,
Kirtland Air Force Base,
Albuquerque, USA

***Abstract*—This paper proposes two methods to maximize the coverage of distinct ground users by an unmanned aerial vehicle (UAV) -mounted mobile base station: a deep reinforcement learning (DRL) approach, and a reward-based greedy approach. The performance of the proposed methods are evaluated based on two aspects: the number of distinct ground users covered, and the delay experienced by a user until it first receives coverage. The distribution of ground users is modelled as a Gaussian Mixture Model (GMM) with fixed and time-varying means with the latter scenario mimicking the mobility of users. Simulation results show that both proposed methods lead to efficient coverage and latency performance with the DRL based approach significantly outperforming the rewards-based greedy algorithm, especially when ground users are allowed to be mobile.**

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

I. Introduction

The use of Unmanned Aerial Vehicles (UAVs) for civilian applications, in particular, using UAVs as portable Mobile Base Stations, has gained traction in the past few years [1] [2]. For this, UAVs are fitted with transceivers and are flown over areas where setting up traditional Base Stations (BSs) may not be feasible [3] [4]. For example, areas that are affected by natural (e.g., floods) or man-made disasters (e.g., terrorist attacks). Unlike traditional BSs, the portability of UAVs allows for choosing a flight path that covers as many distinct ground users as possible.

Several approaches have been proposed over the past years to provide better coverage to ground users. For example, the authors of [5] discuss deployment of single and multiple UAVs to provide coverage to ground users in a way that maximizes fairness and reduces interference. Using Reinforcement Learning (RL) based methods, the authors were able to significantly increase the fairness compared to baseline methods. However, [5] assumed the distribution of users to be static, which may not be the case in practice. Authors of [6] formulated the UAV trajectory optimization problem as a Mixed Integer Linear Programming Problem and proposed an algorithm that finds an optimal trajectory iteratively. The authors assumed that the users are non-stationary and periodically send their location information to the UAV. Based on this information, the algorithm determines an optimal path. In this case, however, the performance and the optimality of the trajectory thus obtained is contingent upon the accuracy of location information relayed to the UAV by the users. Authors of [7] discussed techniques for optimal placement of a UAV in 3D space for maximal coverage of users. An algorithm was developed to determine the optimal location to place the UAV to cover maximum number of users at the required signal to noise ratio (SNR). However, owing to the limited coverage radius of the UAV, it is unlikely that all users can be covered from a single location, especially if users are allowed to be mobile. Placement of a UAV to optimize the user coverage by taking user mobility also into account was discussed in [8]. The authors modeled the movement of users as a random-walk. The distance travelled by a user in each transition of the random-walk is modeled to be Rayleigh distributed with a fixed shape parameter. The position of the UAV is updated periodically, which is solved iteratively based on the temporal coverage probability that the authors derived analytically. However, when the number of users is large, finding the optimal location to place the UAV for each update instance iteratively may not be feasible nor desirable. The authors of [9] proposed an algorithm to minimize the average distance between users and a UAV. When a few users are located outside of a group or a cluster, however, the resulting deployment might result in reduced signal quality for rest of the users due to the outliers.

Most of the papers in literature assess the performance of the system based solely on the number of users covered [6] [7]. This may not be the best metric because it does not take into account the coverage fairness; i.e., it is not possible to determine if a user, or a group of users, have higher uptime compared to other users. In addition, the delay experienced by a typical user until it first received the UAV coverage might also be important. The authors of [5] and [8] take the fairness into account, but latency experienced by users is still not considered. In this paper, we propose two methods to find a UAV trajectory that maximizes the coverage for distinct users while also ensuring fairness in coverage: a deep reinforcement learning (DRL) -based approach, and a reward-based greedy method. We use the complimentary cumulative distribution functions (ccdfs) of number of distinct users covered, as well as the delay experienced by a user until it has coverage for the first time to gain deeper insights on the performance beyond averages. In addition, the performance of our methods are investigated for two different types of user distributions. In the first case, we assume that the users are clustered around regions of high user density called hotspots. In the second case, we allow these clusters to be mobile to model more realistic application scenarios.

The remainder of the paper is organized as follows. Section II discusses our system model and assumptions. Section III discusses the proposed trajectory learning methods. Section IV discusses the performance of the proposed approaches compared to other alternatives in simulated application scenarios. Finally, section V concludes the paper.

II. System Model and the Problem Formulation

A. System Model

We consider a square geographical area of side-length L divided into M square cells of equal size as shown in Figure 1. The number of square cells into which the area is to be divided is determined by the application context (which will be dependent on parameters such as the communications radius of the UAV). An arbitrary but known maximum number of users are assumed to be distributed in 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. For simplicity, the UAV is restricted to hover only over the center and corners 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. This means that with the proposed model, the area of interest need not necessarily be a square. It can be of any arbitrary shape, but with sufficient number of hovering points so that any location can be covered by the UAV from one of the hovering points. With this model, the movement of the UAV can be restricted to nine directions namely: North, North-East, East, South-East, South, South-West, West, North-West or be stationary.

Since UAVs are powered by batteries, their flying time is limited. We divide the total flying time of the UAV into slots of equal duration. During each slot, the UAV is expected to hover over one of the possible hovering points providing connectivity to the users inside its coverage radius. The objective of this paper is to design methods that find a flight path for the UAV-mounted mobile BS to provide coverage to as many *distinct* ground users as possible before exhausting the UAV's energy.

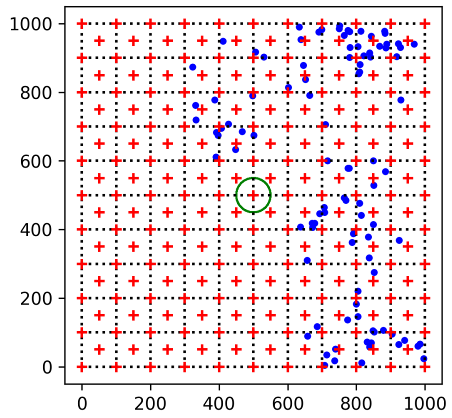

Fig. 1. A UAV hovering at location (500,500). The UAV coverage radius is represented by the green circle. The blue dots indicate the users and the 221 red crosses represent the possible hovering points.

B. User Distribution

The authors of [5] assumed that the users are distributed uniformly over the square area. In reality, however, this may not be the case, since users are usually clustered around places like offices, universities and residential areas. The authors of [10] used the map of downtown San Francisco for planning the path of a UAV. Although this approach is realistic, it may not necessarily generalize well to other places.

In this paper, we use two approaches to model how users are distributed over the area of interest. In the first method, we assume that the users are clustered around certain fixed hotspots. In the second approach, we assume that the users are clustered and the cluster means are time-varying. In both approaches, we model the clusters as Gaussian mixture models (GMMs). Each component or cluster in the mixture is parameterized by cluster weight, π_k, cluster mean, μ_k, and variance, σ_k^2. Thus, the location distribution of the users can be written as:

$$(X_n, Y_n) \sim \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_x, \sigma_x^2), \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_y, \sigma_y^2) \right)$$

where K is the number of clusters and $\sum_{k=1}^{K} \pi_k = 1$. The cluster weights represent the probability of users being in each cluster. For example, Figure 1 corresponds to a user distribution with $K = 4$ clusters.

The cluster mean, $\mu_k = [\mu_x, \mu_y]$, represents the mean X and Y coordinates of a user in each cluster k while the cluster variances, $\sigma_k^2 = [\sigma_x^2, \sigma_y^2]$, represents the location spread along X and Y coordinates. A smaller variance represents a tightly packed cluster whereas a larger variance means a cluster in which users are more spread out. Figure 1 corresponds to a distribution in which all clusters have the same variance of 100.

In the second approach, we allow the cluster means to move in a fixed direction and magnitude. This can be appropriate when the UAV is providing coverage to on-the-move ground troops or people commuting to work from their homes. Note that, the methods developed do not require the user mobility to have a fixed direction and/or magnitude. This is assumed for the purpose of training simplicity. Future work will investigate the scenarios with arbitrary mobility directions and magnitudes.

III. Proposed UAV Trajectory Learning Methods

A. Proposed Method 1: RL Approach Using DQN

According to our model, the total flying time of the UAV is divided into time slots and during each time slot, a decision to be made with regard to the UAV's movement. This setup can be modelled as a Markov Decision Process (MDP). In an MDP, we have an environment and an agent that interacts with the environment. At each time step, t, the agent observes the state of the environment at that time denoted by 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}. The reward function is used to measure an action qualitatively. An action with favorable impact on the performance will have a higher reward whereas an action with an adverse affect on the performance will have a lower reward. The reward function for the t-th time instance is defined as:

$$r_t = (|N_t| - |N_{t-1}|) + g * V + F_t + p_t \tag{1}$$

where N_t is the set of users covered until t-th time instant. Thus, $|N_t| - |N_{t-1}|$ is the number of new users who are not previously covered, $V \in \{0, 1\}$, denotes whether the current hovering point at $(x_t,\, y_t)$ was visited previously, and g is a scaling factor. The Jain's fairness function F_t at time step t for N ground users is defined as [11]:

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

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:

$$Cov_n^t = \begin{cases} 1, & \text{if the user is inside coverage area of UAV} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The penalty function, p_t, is non-zero only if the UAV attempts an action that will cause it to fly out of the designated area:

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

In Eq. (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 results 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 only the number of users covered was taken as the reward function in Eq. (1), there is a possibility that the UAV may get stuck at a hovering point and not move further, because any movement may result in a reduced reward. The reward function, Eq.(1), avoids this because if the UAV hovers at the same location twice, then the first and the second term of the reward function will be zero, thus reducing the reward for those time instants. The proposed reward function encourages the UAV to visit new hovering points, thereby increasing the probability of covering new users.

For any given state S_t, the agent takes an action a_t that maximizes the sum of discounted expected future rewards [12]. In the case of Q-Learning, this is done by maintaining a Q-table, whose entries represent these expected future rewards if a particular action was taken when in a particular state. Hence, the agent selects an action that has the highest expected future rewards as: $a_t = \text{argmax}_a\ Q(S_t,\, a)$.

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 [13], instead uses a deep neural network (DNN) as a function estimator to learn $Q(S_t,\, a)$. The DQN makes use of an experience replay [13] where a replay memory is used to store tuples of transitions consisting of the state, S_t, the action a_t taken at that state, reward r_t obtained by taking that action, and the resulting new state, S_{t+1}. At every time step, the DQN is trained by sampling a random batch of experiences from the experience replay memory. The Algorithm 1 details the DQN-based RL approach.

At any given time instant, the DQN makes a decision based on the current state, S_t. Hence, the state S_t must be selected carefully to incorporate vital details of the environment that can impact the performance. The state S_t for time step t in the proposed method consists of the X and Y coordinates of the location of the UAV at that time instance, the users covered at that location, a binary value indicating whether the current hovering point was visited previously and the remaining UAV energy at that time instant along with the same information for a finite number of previous time instances.

B. Proposed method 2: An AI based greedy UAV trajectory learning 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 defined in Eq. (1). The greedy algorithm does not take into account whether the selected action will have unfavorable results in the long run. It should be noted that if there are two or more actions that yield the same reward, an action is selected at random out of the multiple actions resulting in the identical reward.

Although the algorithm is relatively simple compared to the DQN-based approach proposed in the previous section, it can nevertheless be effective in many situations. Indeed, as we will show in the next section, it was able to perform reasonably well compared to the DQN method with significantly lower computational complexity.

Algorithm 1 DQN Algorithm

```
1:  Initialize policy network $Q(s, a)$ with random weights
2:  Set target network $Q'(s, a)$ with the same weights
3:  Set $\varepsilon = 1$
4:  Create a replay memory
5:  for episode = 1 to M do
6:      for time = 1 to N do
7:          Generate a random number $w$ between 0 and 1
8:          if $\varepsilon > w$ then
9:              Take a random action $a_t$
10:         else
11:             Take action: $a_t = \text{argmax}_a Q(S_t, a)$
12:         Store transition: $(S_i , a_i , r_i , S_{i+1})$ into buffer
13:         Sample random batch of transitions from the buffer
14:         if Episode ends at $i + 1$ then
15:             Set $y_i = r_i$
16:         else
17:             Set $y_i = r_i + \gamma \max_{a'} Q'(S_{i+1}, a')$
18:         Fit the policy network on the data: $(S_t, y_t)$.
19:         Reduce the value of $\varepsilon$
20:         Every K steps, set weights of target equal to policy
    network
```

IV. Simulation Results

We consider an area of $1\,km \times 1\,km$ divided into $M = 100$ square cells of similar size resulting in 221 hovering points as was shown in Figure 1. The users were distributed over this area according to a Gaussian Mixture Model (GMM) with either $K = 4$ or $K = 8$ clusters. The parameters of the GMMs are given in tables I, II and III.

We use three metrics to assess the performance of our proposed methods: (1) the average number of distinct users covered in an episode, (2) the complementary cumulative distribution function (ccdf) of the number of users covered, and (3) the average delay experienced by a user until it receives coverage for the first time (in an episode) where the average delay is computed as:

$$\text{Average Delay} = \frac{\sum_{t=1}^{T}(t \times U_t)}{\sum_{t=1}^{T} U_t}$$

where U_t denotes the number of users covered during time step t, and T denotes the time step when the UAV has exhausted its energy.

Note that, sometimes the ccdf may give better insight on the performance than just comparing the averages. Specifically, the ccdf shows the probability of covering more than a given number of users during a UAV flying episode, which cannot be deduced by knowing only the average number of users covered.

A. Stationary user distributions

In this setup, during each epoch, 100 users are independently and identically distributed (iid) over the desired area according to a GMM with K clusters. An epoch is defined as one run of the UAV from the beginning to exhaustion of its energy. It should be noted that, the number of clusters, and the means and variances of the GMM are fixed throughout each simulation.

With stationary distributions, two scenarios are discussed: (1) clusters with larger variances representing user distributions that are spread out, and (2) clusters with smaller variances representing densely packed users. The parameters of the GMM used in each of the cases are tabulated in Table I and Table II, respectively for the case $K = 4$. Table IV and Table V give additional parameters about the environment and specifications of the Convolutional Neural Network (CNN) used.

TABLE I Gaussian Mixture Parameters for Larger Variance

Number of clusters (K)	4
Cluster Weights (π_k)	0.25,0.25,0.25 and 0.25
Cluster Means (μ_k)	(500,800) (800,0) (800,400) (850,900)
Cluster Variances (σ_k^2)	100, 100, 100 and 100

TABLE II Gaussian Mixture Parameters for Smaller Variance

Number of clusters (K)	4
Cluster Weights (π_k)	0.25,0.25,0.25 and 0.25
Cluster Means (μ_k)	(300,200) (800,400) (200,600) (500,800)
Cluster Variances (σ_k^2)	10, 10, 10 and 10

TABLE III Gaussian Mixture Parameters for Time Varying Means

Number of clusters (K)	4
Cluster Weights	0.25,0.25,0.25 and 0.25
Cluster Means	(300,200) (800,400) (200,600) (500,800)
Cluster Variances	100, 100, 100 and 100
Direction	$\pi/4$ radians
Magnitude	$1m$

The DQN algorithm learns iteratively by means of trial and error. It takes about 500 training epochs for the algorithm to find an optimal path as can be inferred from Figure 2.

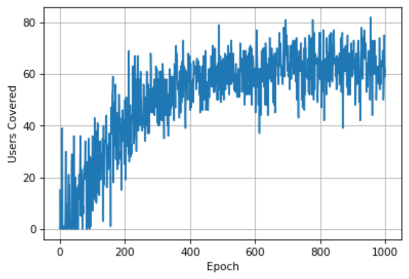

Fig. 2. Epoch as a Function of the Number of Users Covered

Figure 3 shows the number of distinct ground users covered by the reward-based greedy, the trained DQN and random action selection algorithms when users are distributed according to GMMs with $K = 4$ and $K = 8$. Note that, the latter takes random decisions at each time step. From Figure 3, it can be seen that compared to random action selection method, a significant number of users are covered by the proposed methods. Moreover, it is seen that the proposed rewards-based greedy algorithm performance is only about 15% - 20% less than that of the DQN-based approach. It is also interesting to notice that the performance of both learning methods slightly deteriorates when the number of clusters increases while the opposite is true for the random action selection algorithm. This may be attributed to the fact that more clusters make systematic learning more difficult while helping random guesses to be more successful.

TABLE IV Simulation parameters

Area	1000m x 1000m
Number of Blocks (M)	100
Size of Blocks	100m x 100m
Number of Hovering points	221
Number of users (N)	100
Number of time slots	67
Radius of coverage of UAV	50m
Penalty (P)	-1
g	0.2

TABLE V Specifications of the Convolutional Neural Network used as the DQN

Input Shape	10x5x1
Kernel Size	2 x 2
Padding	Same
Layers	3
Number of filters in each layer	40, 45 , 55
Activation function	Sigmoid
Optimizer	Adam
Loss Function	Huber
Gamma	0.9
Learning rate	0.0001

Let U denote the number of distinct users covered by the UAV in an epoch. In Figure 3 we show the ccdf of U given by $Pr(U > n)$. It is worth observing form Figure 3 that the maximum number of users that can be covered with a non-zero probability is also larger with the DQN based trajectories compared to that with the greedy trajectories.

Figure 4 shows the cumulative distribution function (cdf) of average delay until a user is first covered in an episode. Clearly, both proposed learning algorithms result in much smaller average delays than with the random actions. Perhaps surprisingly, the average delay with the greedy algorithm is not too far from the DQN-based learning algorithm, again showing that the proposed rewards-based greedy algorithm is a good alternative to the DQN-based learning at much less computational complexity and learning duration.

In some practical scenarios, users may be densely concentrated around mobile hotspots like offices, universities, or residential areas. Figure 5 and Figure 6 show the ccdf of number of users covered and the cdf of average delay seen by a user for the first time coverage during an epoch for this case, respectively. It can be seen from Figure 5 that the proposed DQN and the rewards-based greedy methods are at par with each other, covering more than 70 and 50 users on average in four and eight cluster cases, respectively. Moreover, Figure 5 shows that DQN-based learning offers additional desirable performance traits. For example, it can be seen from Figure 5 that the greedy algorithm is not able to find and provide coverage to all 100 users with non-zero probability and, in fact, the maximum number of users covered with non-zero probability can be far less than 100. However, the DQN-based algorithm is able to find most of the users with non-zero probability. In the case of $K = 4$ clusters, Figure 5 shows that it is able to find all 100 users with more than 0.4 probability!

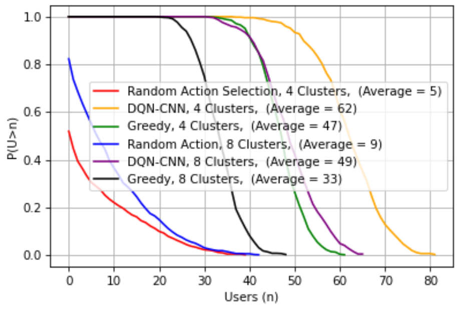

Fig. 3. Complementary Cumulative Distribution Function Showing Probability of Covering More Than n Users

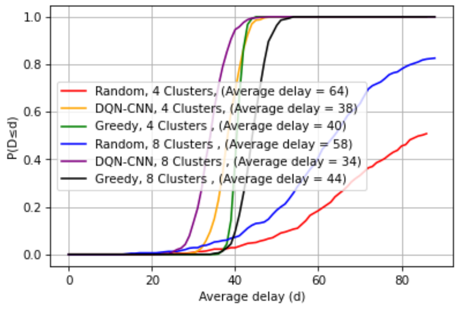

Fig. 4. Cumulative Distribution Function Showing Probability of the Average Delay $< d$

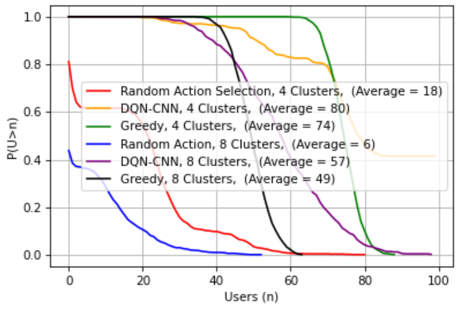

Fig. 5. Complementary Cumulative Distribution Function Showing Probability of Covering More Than n Users

B. Non-stationary user distributions

In scenarios where ground users are mobile, like troops on the move or users commuting to work, it makes sense to model

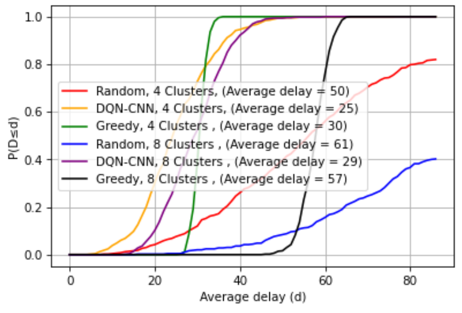

Fig. 6. Cumulative Distribution Function Showing Probability of the Average Delay < d

the users as clusters, but with moving means. The parameters for the GMM in this case is given in Table 3. For simplicity, the directions and magnitudes of movements are assumed to be fixed.

Since the distribution of users at a time instance is correlated to distribution of users at previous time instances, for this scenario we may expect a Recurrent Neural Network (RNN) to be a better candidate for the DQN over CNNs, since RNNs are known for their ability to capture the temporal correlations in the input.

Figure 7 and Figure 8 illustrate the performance of the proposed and random action selection methods in this scenario. It can be inferred from Figure 7 that the proposed DQN method covers significantly more users than the reward-based greedy method. Moreover, the proposed DQN method with RNN as the function estimator is slightly better than the one with CNN, which can also be inferred from ccdf plots shown in Figure 7. Furthermore, the average delay experienced by users was significantly lower when using the proposed DQN method in contrast to random action selection method and reward-based greedy method, which can be seen in Figure 8. In fact, the DQN method with RNN resulted in fewer delays in the case of $K = 4$ clusters.

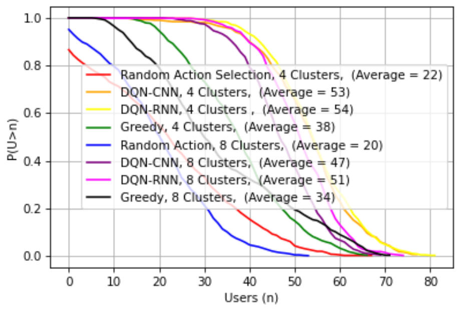

Fig. 7. Complementary Cumulative Distribution Function Showing Probability of Covering More Than n Users

V. Conclusion

This paper proposed DRL and reward-based greedy methods to maximize the coverage of distinct ground users by a UAV-mounted base station. The performance of the proposed methods were analyzed based on the number of users covered and the delay experienced a by user until it first receives coverage while assuming two scenarios for the distribution of users: (1) a GMM with fixed and (2) time-varying means. It was observed that the proposed methods were able to cover a significantly more number of users compared to random actions in both cases. Computationally lightweight greedy algorithm was observed to perform very close to the DRL method when user distributions were stationary. However, the DRL was shown to outperform the greedy algorithm significantly, especially when users were allowed to be mobile. Further work is needed to incorporate complex mobility models for user distribution and also to take into account wireless channel effects.

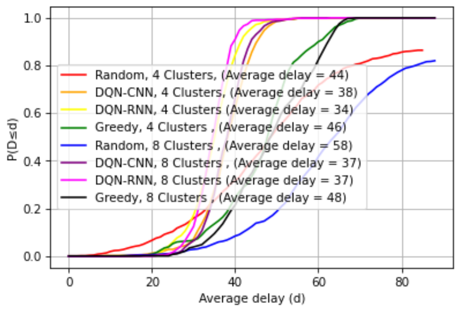

Fig. 8. Cumulative Distribution Function Showing Probability of the Average Delay < d

References

[1] M. Böyük, R. Duvar, and O. Urhan, "Deep Learning Based Vehicle Detection with Images Taken from Unmanned Air Vehicle," in *2020 Innov. Intell. Syst. Appl. Conf. (ASYU)*, 2020, pp. 1–4.

[2] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2624–2661, 2016.

[3] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless Communications with Unmanned Aerial Vehicles: Opportunities and Challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 36–42, 2016.

[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," in *2016 IEEE Int. Conf. Commun. (ICC)*, 2016, pp. 1–5.

[5] 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," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 67–81, 2020.

[6] G. Li, C. Zhuang, Q. Wang, Y. Li, X. Xu, and W. Zhou, "A UAV Real-time Trajectory Optimized Strategy for Moving Users," in *2019 11th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2019, pp. 1–6.

[7] N. Cherif, W. Jaafar, H. Yanikomeroglu, and A. Yongacoglu, "On the Optimal 3D Placement of a UAV Base Station for Maximal Coverage of UAV Users," in *GLOBECOM 2020 - 2020 IEEE Global Commun. Conf.*, 2020, pp. 1–6.

[8] M. Peer, V. A. Bohara, A. Srivastava, and G. Ghatak, "User Mobility-aware Time Stamp for UAV-BS Placement," in *2021 IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2021, pp. 1–6.

[9] A. V. Savkin and H. Huang, "Deployment of Unmanned Aerial Vehicle Base Stations for Optimal Quality of Coverage," *IEEE Wireless Commun. Lett.*, vol. 8, no. 1, pp. 321–324, 2019.

[10] V. Saxena, J. Jaldén, and H. Klessig, "Optimal UAV Base Station Trajectories Using Flow-level Models for Reinforcement Learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1101–1112, 2019.

[11] R. Jain, D. M. Chiu, and H. WR, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," *CoRR*, vol. cs.NI/9809099, 01 1998.

[12] R. S. Sutton and A. Barto, *Reinforcement learning: An Introduction*. MIT Press, 2018.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602