

UAV Access Point Placement for Connectivity to a User with Unknown Location Using Deep RL

Enes Krijestorac, Samer Hanna, Danijela Cabric

Electrical and Computer Engineering Department,

University of California, Los Angeles

enesk@ucla.edu, samerhanna@ucla.edu, danijela@ee.ucla.edu

Abstract—In recent years, unmanned aerial vehicles (UAVs) have been considered for telecommunications purposes as relays, caches, or IoT data collectors. In addition to being easy to deploy, their maneuverability allows them to adjust their location to optimize the capacity of the link to the user equipment on the ground or of the link to the basestation. The majority of the previous work that analyzes the optimal placement of such a UAV makes at least one of two assumptions: the channel can be predicted using a simple model or the locations of the users on the ground are known. In this paper, we use deep reinforcement learning (deep RL) to optimally place a UAV serving a ground user in an urban environment, without the previous knowledge of the channel or user location. Our algorithm relies on signal-to-interference-plus-noise ratio (SINR) measurements and a 3D map of the topology to account for blockage and scatterers. Furthermore, it is designed to operate in any urban environment. Results in conditions simulated by a ray tracing software show that with the constraint on the maximum number of iterations our algorithm has a 90% success rate in converging to a target SINR.

Index Terms—UAV, relay, IoT, reinforcement learning

I. INTRODUCTION

Due to their high mobility and low cost, unmanned aerial vehicles (UAVs) have found their way to many applications in recent years, including package delivery, law enforcement, search and rescue, etc. Following this trend, UAVs are getting an increased attention in the telecommunications sector. Deploying UAVs as aerial basestations has recently emerged as an idea to respond to high localized traffic demands in the next-generation cellular networks [1]–[3]. Using UAVs in such way provides the opportunity to exploit their agility of motion to improve the air-to-ground link capacity by optimal air placement. UAVs can also be utilized for data harvesting in IoT or as data caches and in these applications it is also important to maximize the air-to-ground capacity by optimal placement.

In this paper, we are interested in optimizing the capacity of the channel between the UAV and a ground user in an urban environment. This is a challenging problem considering that the environment between the UAV and the ground equipment can be abundant in scatterers and therefore hard to account for analytically and numerically. Nevertheless, the said problem has been addressed in literature before, under different

assumptions. In most cases, however, the solutions are based on the assumptions that the ground user locations are known and that the wireless channel can be predicted with a simple model.

The problem of a UAV relay placement has been considered mostly for line-of-sight (LOS) channels. In [4]–[6] transmit power and placement of a UAV relay are jointly optimized. However, the authors' solutions apply only to a LOS propagation channel, which makes this approach less applicable in the environments with scattering and obstacles, such as cities. Furthermore, all of [4]–[6] assume that the locations of the users and channel propagation characteristics are known. In [7], a method for optimizing the location of a ground unmanned vehicle is proposed. The approach relies on user location and assumes a fading channel. The algorithm predicts the channel quality across the entire map from a small number of measurements and then using stochastic dynamic programming an unmanned vehicle is optimally routed. While this paper considers ground vehicles, it is relevant to our work since their approach can apply to aerial vehicles flying at a fixed altitude.

In addition to statistical models, some of the previous approaches have also utilized topology maps. The work in [8] considers a UAV swarm that relays communication between users on the ground in an urban environment. The approach relies on known user locations and topology maps to perform swarm particle optimization of placement. Works [9]–[11] utilize 3D topology maps to help UAV placement optimization. Additionally, [9] uses a statistical channel model and the user location to perform optimization. While [10] does rely on user locations, it does not need to know the channel model parameters as these are learned. The algorithm in [11] simultaneously learns user locations and parameters, but as a result, the approach has an extensive learning phase.

Seeking to develop an exploration algorithm that performs learning and placement optimization simultaneously we turned to reinforcement learning. Reinforcement learning has been previously applied to similar problems. [12] uses table-based Q-learning for optimal placement of aerial basestations with the knowledge of user locations. However, since the inputs to this algorithm are only user and UAV locations, it cannot perform outside of the environment it has been trained on. Similarly, [13] uses received signal strength at the UAV and the UAV location to track indoor users with a shallow Q-

This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

learning algorithm. However, the paper only considers two indoor scenarios and the training and testing dataset are the same. Therefore, it is not clear whether the algorithm could perform in a new environment.

We address the problem of optimal UAV placement assuming that the user location is not known. Algorithms that rely on statistical models of the channel may fail to generalize to all environments since the local topology can significantly differ from statistical predictions. To that end, we use deep reinforcement learning to obtain a model-free algorithm for UAV positioning. The proposed algorithm relies on the knowledge of local topology and does not require the knowledge of the user location. Deep reinforcement learning allows us to take a high-dimensional input that is the topology map and use it alongside SINR measurements collected on the trajectory of the UAV to predict the optimal direction of motion. We test our performance in a realistic environment that emulates the wireless channel using a ray-tracing software. Furthermore, the training and the testing dataset are different.

The rest of this paper is organized as follows. In Section II we introduce the relevant reinforcement learning background, define our problem and describe how reinforcement learning can be used to tackle it. In Section III, we describe the simulation environment. Sections IV and V are dedicated to results and conclusions, respectively.

II. UAV PLACEMENT USING Q-LEARNING

In this section, we first introduce the necessary background in reinforcement learning in part II-A. Next, in subsection II-B, we formulate our problem as a partially observable Markov decision process (PO-MDP) and in subsection II-C we discuss how we apply Q-learning to this PO-MDP.

A. Reinforcement learning background

Reinforcement learning (RL) is the branch of machine learning that is concerned with making sequences of decisions. It is mainly concerned with problems that can be casted as a Markov decision process (MDP). In an MDP, an agent \mathcal{A} is situated in an environment \mathcal{E} . At each timestep t , the agent is in a state s_t and takes an action a_t , it receives a reward r_t while moving into the state s_{t+1} . In a partially observable MDP (PO-MDP), the full knowledge of the state of the environment is not known to the agent and in that case it will only have access to an observation of the state. This observation then replaces the function of the state in the reinforcement learning algorithms.

The objective function in reinforcement learning is often the expectation of the discounted reward, $\mathbb{E} \sum_{t=0}^{\infty} \gamma^t r_t$, where γ is the discount factor. Reinforcement learning methods can broadly be classified into two categories: policy learning and Q-learning. In policy learning methods, the goal is to learn the optimal policy function that defines $\pi(a|s)$, which is the probability of taking the action a in the state s . The policy is often deterministic and in that case $\pi(a|s)$ defines a single

action in each state. In Q-learning methods, the goal is to learn the Q-value function

$$Q(s, a) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right],$$

that defines the expected reward in a state s , after taking the action a . If the Q-value function is known, the optimal action in a state s is then $\operatorname{argmax}_a Q(s, a)$. Deep Q-learning is an extension to the Q-learning paradigm that uses deep learning models, such as convolutional neural networks and recurrent neural networks to approximate $Q(s, a)$. Furthermore, Q-learning is a model-free reinforcement learning technique, meaning that it does not rely on known dynamics of the system.

One of the first deep Q-learning algorithms was proposed by Deepmind and was successfully demonstrated on Atari video games [14]. The algorithm was named the deep Q-network (DQN) algorithm. In it, the Q-value function is parametrized by a neural network Q_θ , with parameters θ . An estimate of the true Q-value at time t , Q_t , can be obtained by using a single sample estimate of the Bellman backup operator

$$\widehat{T}Q_t = r_t + \max_{a_{t+1}} \gamma Q_\theta(s_{t+1}, a_{t+1}) \quad (1)$$

This is called a single sample estimate because only the reward r_t at the current time instant t is used to approximate the infinite horizon Q-value function.

In order to approximate Q by Q_θ the following minimization is done over sample data,

$$\operatorname{minimize}_\theta \sum_t \left\| \widehat{T}Q_t - Q_\theta(s_t, a_t) \right\|^2 \quad (2)$$

In the DQN algorithm, the training and the interaction of the agent with the environment happen in parallel. As the agent gathers experience, samples of that experience are stored and the minimization in the Equation 2 is done periodically, every τ_L steps, by randomly sampling a batch of B_L recorded samples and applying gradient descent. This is referred to as experience replay. Samples are arrays of data (s_t, a_t, r_t, s_{t+1}) and these are stored in the replay buffer.

The agent interacts with the environment following the ϵ -greedy policy, where at any time t the agent either takes a random action at probability ϵ or the Q-value optimal action $\operatorname{argmax}_a Q_\theta(s, a)$ at probability $(1 - \epsilon)$. Over the course of the training, the value of epsilon decreases from 1 to 0. In the implementation of DQN, there is an additional Q_θ , called the target Q-network. The target Q-network is used in the Bellman backup operator but it is not optimized over. Instead, it is periodically copied from the main Q-network. The target Q-network is included to improve the stability during training.

The original vanilla DQN algorithm has been improved upon over the years. The two expansions that we will use are double Q-learning [15] and dueling networks [16]. For the sake of brevity we omit the details of these algorithms and the reader is referred to the cited works for more information.

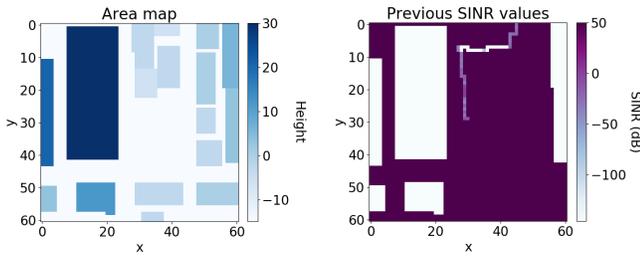


Fig. 1. Example observation for an agent moving according to a random exploration policy. Note that the SINR values of areas that are obstructed and therefore unreachable are set to a very low value (-150 dB), while the areas that have not yet been visited are set to a very high value outside of the possible range of SINR values (50 dB).

B. UAV placement problem as a PO-MDP

We now formulate the UAV placement problem as PO-MDP. We consider the scenario of a UAV located in an urban environment communicating to a radio device on the ground. The area topology is such that LOS connection to the ground user is not always possible, and the communication will often occur over non-line-of-sight paths. At time zero, the UAV and the ground device are located at random positions and the goal of the UAV is to adjust its position so as to increase the SINR at the UAV. We assume that the UAV receives some signal power from the user on the ground to begin with. The UAV moves until an SINR threshold is reached or until maximum time for optimization expires.

In the following, we describe the mechanics according to which the UAV moves around. Since exploring the entire 3D space is a complex task, we restrict the motion of the UAV to the horizontal plane and assume that its altitude is kept constant. We do this as a relaxation but it is worth pointing out that the optimal position for a UAV will often be at the lowest allowed altitude since this brings it closest to the ground user. The UAV can only move around buildings or fly above buildings that are below its altitude and it makes adjustments in its position at discrete time steps. We restrict the directions of the motion of the UAV to the four orthogonal horizontal directions and the motion step size d_S is fixed. We impose this constraint because Q-learning lends itself better to tasks with a discrete set of actions. With these restrictions on the motion, the UAV effectively moves in a uniform plane grid space that spans the environment.

We assume that the UAV location is known. Furthermore, the UAV has access to a 3D map of the environment that maps all the buildings, which can be drawn from a database. 3D maps of major urban areas are generally available and easy to acquire. We also make the assumptions that the channel is slow fading and that the user location does not change significantly over the course of optimization. Furthermore, we assume that there is a sufficient backhaul capacity between the UAV and the core basestation, so we only focus on optimizing the capacity between the UAV and the ground device.

1) *Observation space*: Since the full state of the system in which the UAV operates is not available, the agent in

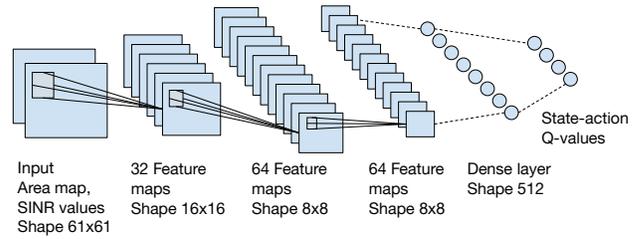


Fig. 2. Neural network model used as the Q-network.

our algorithm relies on two types of observations of the environment to drive its decision making. The first type of observation is the 3D map of the local area. The local area in our case is a square area of side l_O centered at the current location of the UAV. The 3D map information is compressed into a 2D array representation, where each entry represents a grid point in the local area and the value of each entry corresponds to the height of the terrain at that point relative to the UAV altitude. We use heights relative to the UAV altitude to make the algorithm adjustable to different starting UAV flying heights.

The second type of observation that the agent uses are the SINR measurements at previously visited locations in the local area. These are also stored in a 2D array with grid point locations matching the locations of the grid points in the topology observation. The value of each entry is the measured SINR value. To complete the array, we populate the entries with unknown SINR values with a high value P_H outside of the regular range. Furthermore, the points that are blocked by buildings and cannot be visited are populated with low values P_L outside of the span of possible SINR values. With successful training, the algorithm will learn the significance of P_H and P_L . Example observations are shown in Figure 1.

2) *Action space*: Since the action is the motion of the UAV at each time step, it can take on the values $(0, d_S)$, $(0, -d_S)$, $(d_S, 0)$, $(-d_S, 0)$, where each vector represents the displacement in the x- and y-coordinates.

3) *Reward*: The UAV receives a reward equal to the difference between the SINR in the next state and the SINR in the current state. This incentivizes the agent to move towards higher SINR. Furthermore, we assign a constant exploration reward c_E which the agent receives for visiting a new location. We empirically established that an exploration reward incentivizes the agent to explore further away from its starting location, which results in better performance. The reward is mathematically expressed as

$$r_t = \text{SINR}_{t+1} - \text{SINR}_t + c_E \delta_t^E,$$

where δ_t^E is an indicator function activated when the UAV visits a new location.

C. Deep Q-learning implementation

With our problem casted as a PO-MDP, we can apply the deep Q-learning algorithm. For our application, we utilize double Q-learning [15] and dueling networks [16] extensions

to the base DQN algorithm. The choice of the neural network model used as the Q-network depends on the application and therefore it needs to be carefully selected for optimal performance. The neural network model we used is shown in Figure 2. At the input, there are two 2D arrays, corresponding to the SINR and topology observations described in the previous subsection. As displayed, we use 3 convolutional layers with varying number of filters and with each layer having a different filter size. There are two fully connected layers, with the final layer output corresponding to the Q-value for each of the possible actions. We use ReLU as the activation function after each layer prior to the last layer. The layer enabling the dueling networks extension is located before the final layer, however we do not show it in the figure for clearer presentation. Additionally, we used batch normalization and dropout with probability p_D to accelerate the training of the neural network and for regularization purposes.

III. SIMULATION ENVIRONMENT

We use two separate environments for the training and the testing of our algorithm, shown in Figure 3 and Figure 4, respectively. Both spaces are meant to resemble a typical medium-elevation urban area.

In order to create realistic conditions to train our Deep RL model, we used a ray-tracing software called Wireless InSite [17] to emulate the wireless channel. For a given user on the ground we measure the SINR across a uniform grid of points at a fixed height that corresponds to the UAV flying altitude. The grid points are separated by 4 meters and they span the entire environment. The UAV altitude is set to 10 meters. To generate the training data, the user radio was placed at 27 locations uniformly spanning the training environment, while for the testing data we placed the user at 25 different locations in the testing environment. The numbers were decided such that user locations uniformly cover the entire space, while still taking a feasible amount of time to make calculations for in the ray tracing software. The users transmit a narrow-band signal of 20 dBm power using a frequency of 800 MHz. We introduce a Gaussian noise and a Gaussian background interference across the space with the average combined power of -104 dBm. Half-wave dipole antennas with vertical orientation are used at the user and the UAV.

The SINR measurements were then exported and used to build a training and a testing environment in software that the DQN algorithm can interact with. At each realization or episode of the environment we use the SINR measurements for a random user location and the UAV is placed at a random location on the grid. The locations that the UAV can visit correspond to the ones where SINR measurements were recorded. The episode finishes if the UAV reaches the required SINR or the maximum number of steps that the UAV can take is exceeded.

IV. RESULTS

In the first part of this section we describe the details of the training of our algorithm and demonstrate that it learns how to

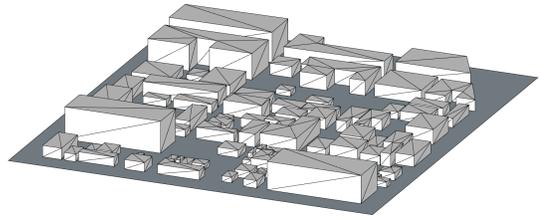


Fig. 3. The training urban environment. Approximate size: 550x500 m.

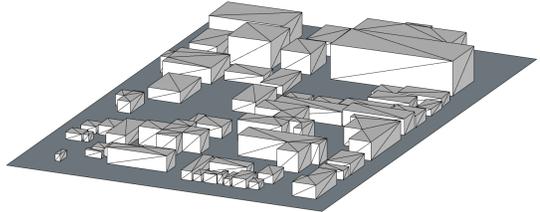


Fig. 4. The testing urban environment. Approximate size: 400x500 m.

move the UAV in order to increase SINR. In the second part, we validate the performance of the algorithm in the testing environment and compare it to a genie algorithm in terms of steps made until convergence to the required SINR.

A. Training

We train the DQN algorithm in the training environment described in the previous section. The maximum number of steps during an episode t_{MAX} was set to 800 and the target SINR P_T , was set to 5 dB. When deploying the UAV on the map we ensure that it is not placed in a dead zone with no signal reception, as this would be outside of our problem statement. In the ray-tracing simulator, these regions occur when there are no direct or reflected paths that can reach the UAV. For regularization purposes, we rotate the coordinate system of the map by a random multiple of 90° every training episode. This ensures that the algorithm does not become biased towards moving in any particular direction over the course of the training.

We use the ϵ -greedy policy for exploration, however the agent's random actions are steered. Namely, the agent never takes a random action that would lead to it leaving the map or colliding with a building. Furthermore, the agent repeats the action it has taken in the previous step at probability 0.4ϵ and takes any random allowed action at probability 0.6ϵ . The repeated movements lead to the agent exploring a larger area through random walk in the early training stage, instead of staying confined to the local space around the starting location. The optimal action $\text{argmax}_a Q_\theta(s, a)$ is taken at probability $1 - \epsilon$. We also ensure that the agent never leaves the map or collides with a building when taking actions according to the DQN. This is done by choosing an action that gives the highest Q-value while still being a legal movement in the environment. As the UAV moves around, its experience samples are stored in a replay buffer that can store up to 5×10^5 samples and

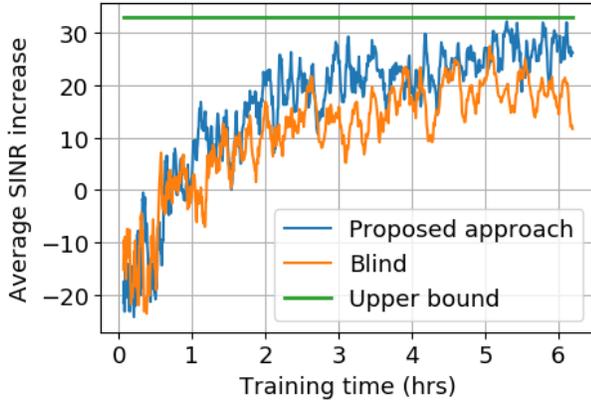


Fig. 5. The training results for the proposed model and blind model that does not rely on topology information.

when this limit is reached the oldest samples are thrown out to make space for the new ones.

The parameter values used for the training of the DQN algorithm are shown in Table I. They were selected after tuning for best performance. During minimization we employ gradient clipping and also anneal the learning rate. The width and length of the observation is 61 grid points or 244 m. The movement step size d_S was 4 m.

The training results are shown in Figure 5. To keep track of the progress of training, we measure the average SINR increase from the SINR at the start of the episode to the SINR at the end of the episode, over the most recent 100 episodes.

To demonstrate the benefits of using 3D maps we evaluate a DQN algorithm that doesn't rely on the 3D map but is otherwise identical to our proposed algorithm. We refer to this algorithm as the 'blind' algorithm. The blind algorithm only has a 2D SINR array as an input and the regions blocked by buildings are not populated by P_L but instead left as P_H . In training, we use the same parameter settings for the blind and the proposed algorithm.

Furthermore, we include an upper bound on the mean SINR increase in the training stage. It is calculated by taking the average of the SINR differences between the SINR at all possible starting UAV locations and P_T , for all user locations. This is an upper bound on the mean performance across a large number of episodes.

The results in Fig. 5 show that the learning capacity of our proposed algorithm is larger than that of the blind algorithm. The intuitive explanation for this is that the algorithm with the knowledge of the topology is more efficient in exploring the

TABLE I
DQN PARAMETER VALUES USED IN TRAINING

Description	Parameter
Exploration reward	$c_E = 1.2$
Discount factor	$\gamma = 0.99$
Training batch size	$B_L = 20$
Training interval	$\tau_L = 3$
Dropout probability	$p_D = 0.4$

space because it can eliminate obstructed areas and because the building knowledge combined with SINR measurements can give it indication where to move to find better SINR. The performance curves are noisy due to the nature of training through experience replay and due to the fact that over a 100 episodes the algorithm only experiences a subset of the training environment, which makes the difficulty vary as some user locations are harder to find optimal paths for than others.

B. Testing

In the testing stage, the algorithm is placed in an entirely new environment and relies only on the trained neural network Q_θ to guide the movement of the UAV. We use the copy of Q_θ that had the highest performance during training. We also introduce a small amount of randomness in decision making, which we found to lead to a better performance. The agent takes a random action at probability 0.096. The value of P_T for the testing case was again set to 5 dB and the maximum number of steps t_{MAX} was reduced to 500, since the testing environment is smaller than the training environment.

In Fig. 6, we show a number of trajectories of the UAV moving according to our algorithm, where the target SINR was reached in less than t_{MAX} steps. The upper figure shows the building height relative to the UAV altitude and the lower figure shows the heatmap of the SINR across the map. The user location is the same in each episode, and we place the UAV at random locations. We can see that the algorithm is capable of following the direction that leads it to improving the SINR and it is also capable of correcting itself when realizing that the current direction of motion is not leading it to better SINR. In the instances where the UAV loops around its location, we can infer that the algorithm explores several path options before settling on the one it deems optimal. Buildings are avoided by the algorithm and it can be inferred that the algorithm eliminates path options because of them. An evidence for the latter is that the UAV tends to move parallel to the building edges, for example.

Finally, we analyze the performance of our algorithm over many realizations. To get a reference on how fast our algorithm converges to an optimal point we used a genie algorithm for optimal placement. The genie algorithm has a complete knowledge of the SINR distribution and building topology, and uses dynamic programming to find the shortest path to a location with sufficient SINR. We run the proposed and the blind algorithm for 1000 realizations across the entire testing data set over different user positions. The results for the number of steps made until convergence to the sufficient SINR for all three algorithms are shown in Fig. 7. The median number of steps until convergence for the proposed algorithm is 44 and 69 for the blind algorithm. Furthermore, the proposed algorithm is 90% successful in under t_{MAX} steps compared to 66% of the blind algorithm. Therefore, we show that the knowledge of topology map assists our algorithm even in a novel environment. The median number of steps required for the genie algorithm is 14. The difference in the number of steps required relative to the proposed algorithm is due to the

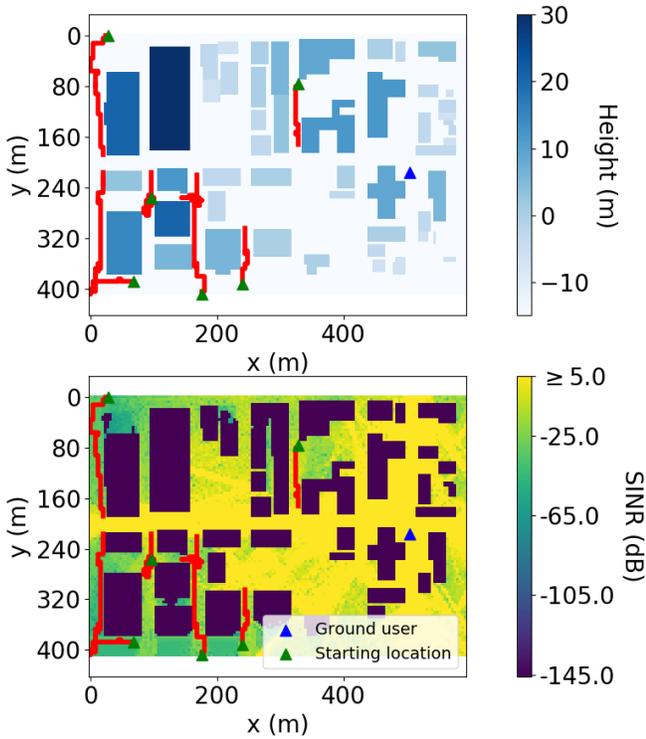


Fig. 6. Successful trajectories of the UAV moving according to our algorithm. The upper figure shows the building height relative to the UAV altitude and the lower figure shows the heatmap of the SINR across the map. The green triangle markers represent the starting positions of the UAV. The blue marker represents the location of the user on the ground.

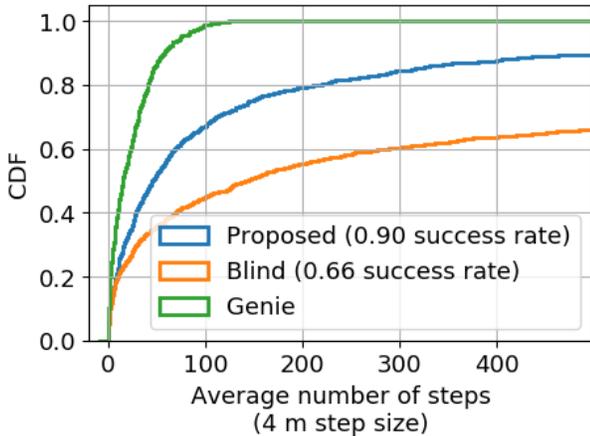


Fig. 7. The CDF of the number of steps until convergence to the sufficient SINR for the proposed approach, the blind approach and the genie algorithm.

fact that our algorithm has to explore the space to find good SINR since it does not where the points with sufficient SINR are a priori.

V. CONCLUSIONS

In this paper, we used deep reinforcement learning methods to optimize the placement of a UAV communicating to the user on the ground. We consider the case where the ground

user location is not known and use topology data to replace statistical models of the channel. We were able to achieve 90% success rate in moving the UAV to a location that has a sufficient SINR within a limited number of steps. Moreover, our reinforcement learning approach stands out in that it can be applied in any urban environment. Our future work will focus on the scenario where the ground user is moving over the course of optimization. Furthermore, we will explore how the deep reinforcement learning techniques can be used to simultaneously optimize the locations of multiple UAVs serving multiple users on the ground.

REFERENCES

- [1] Y. Li and L. Cai, "UAV-assisted dynamic coverage in a heterogeneous cellular system," *IEEE Network*, vol. 31, no. 4, pp. 56–61, 2017.
- [2] Q. Wu, J. Xu, and R. Zhang, "UAV-enabled aerial base station (BS) III/III: Capacity characterization of UAV-enabled two-user broadcast channel," *arXiv preprint arXiv:1801.00443*, 2018.
- [3] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1647–1650, 2016.
- [4] H. Wang, G. Ren, J. Chen, G. Ding, and Y. Yang, "Unmanned aerial vehicle-aided communications: Joint transmit power and trajectory optimization," *IEEE Wireless Communications Letters*, vol. 7, no. 4, pp. 522–525, 2018.
- [5] Y. Jin, Y. D. Zhang, and B. K. Chalise, "Joint optimization of relay position and power allocation in cooperative broadcast wireless networks," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2493–2496, Mar. 2012.
- [6] Y. Zeng, R. Zhang, and T. J. Lim, "Throughput Maximization for UAV-Enabled Mobile Relaying Systems," *IEEE Transactions on Communications*, vol. 64, pp. 4983–4996, Dec. 2016.
- [7] A. Muralidharan and Y. Mostofi, "Path planning for a connectivity seeking robot," in *2017 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2017.
- [8] P. Ladosz, H. Oh, and W.-H. Chen, "Optimal positioning of communication relay unmanned aerial vehicles in urban environments," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1140–1147, IEEE, 2016.
- [9] J. Chen and D. Gesbert, "Optimal positioning of flying relays for wireless networks: A los map approach," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2017.
- [10] O. Esrafilian, R. Gangula, and D. Gesbert, "Learning to communicate in UAV-aided wireless networks: Map-based approaches," *IEEE Internet of Things Journal*, 2018.
- [11] O. Esrafilian, R. Gangula, and D. Gesbert, "UAV-relay placement with unknown user locations and channel parameters," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pp. 1075–1079, IEEE, 2018.
- [12] X. Liu, Y. Liu, and Y. Chen, "Deployment and movement for multiple aerial base stations by reinforcement learning," in *2018 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2018.
- [13] M. M. U. Chowdhury, F. Erden, and I. Guvenc, "RSS-based Q-learning for indoor uav navigation," *arXiv preprint arXiv:1905.13406*, 2019.
- [14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
- [15] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [16] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.
- [17] "Wireless EM propagation software - Wireless InSite." URL: <https://www.remcom.com/wireless-insite-em-propagation-software>.