Deploying UAV Base Stations in Communication Network Using Machine Learning

by

Xukai Zhong
B.ASC., Simon Fraser University, 2017

A Report Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

Deploying UAV Base Stations in Communication Network Using Machine Learning

by

Xukai Zhong
B.ASC., Simon Fraser University, 2017

Supervisory Committee

Dr. Xiaodai Dong, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Wu-Sheng Lu, Departmental Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Xiaodai Dong, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Wu-Sheng Lu, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Today has witnessed a constantly increasing demand for high-quality wireless communications services. Moreover, the quality of service (QoS) requirement of future 5G and beyond cellular networks leads to the possible use of the unmanned aerial vehicle base station (UAV-BS). Deploying UAV-BSs to assist the communications network has become a research direction with great potential. In this project, we focus on the problem of deploying UAV-BSs to provide satisfactory wireless communication services, with the aim that maximizes the total number of covered user equipment subject to user data rate requirements and UAV-BS capacity limit. Then, the report extends to a reinforcement learning based method to adjust the locations of UAVs to maximize the sum data rate of the user equipment (UE). Numerical experiments under practical settings provide supportive evidences to our design.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Dr.Xiaodai Dong who provided me insightful advice and numerous knowledge throughout my graduate study. Besides my supervisor, I would like to thank Dr.Wu-Sheng Lu for acting as my supervisory committee, Dr.Yiming Huo for providing assistant with my project.

Also I would like to thank my parents and my girl friend for supporting me in any conditions, providing help whenever I am in need. I'm also grateful to many of my friends who have brought me happiness, joy as well as generous help, especially Ahmed Elmoogy, Hoang Minh Tu, Dr.Jinlong Zhan, Tong Zhu, Tianzhu Li, Ying Wang and Ji Shi.

DEDICATION

To my parents

Chapter 1

Introduction

1.1 Background

Wireless communications systems which include unmanned aerial vehicles (UAV) are capable of providing cost-effective wireless connectivity for devices without fixed infrastructure base stations. Compared to terrestrial communications or those based on high-altitude platforms, on-demand wireless systems with low-altitude UAVs are in general more flexibly reconfigured, and likely to have better communications channels due to the presence of short-range line-of-sight links [15]. For example, in the extreme situations like natural disaster or battlefield where it is not cost-efficient nor time-efficient to re-deploy onsite terrestrial base stations, the utilization of unmanned aerial vehicle base stations (UAV-BSs) becomes a valid solution since UAV-BSs can be deployed and reconfigured rapidly. Also, the UAV can play an important role in practical applications of Internet of Things (IoT) where UAV collects data from IoT devices [12]. Moreover, UAVs have a great potential to be used in many 5G and beyond applications, for example, the authors in [7] propose a multi-layer UAV network model for UAV-enabled 5G and beyond applications.

With their high mobility and low cost, in the past few decades, UAVs have found a wide range of applications including wireless communications, rescue and agriculture. Historically, UAVs have been primarily used in the military [15], mainly deployed in hostile territory to reduce pilot losses. With the continued reduction of the cost as well as the size of the devices, small UAVs are now becoming more easily accessible to the general public. Therefore, lots of new applications in the civilian and commercial domains have emerged, with typical examples including weather monitoring,

communications relaying, and others.

For practical use of UAV in wireless communications, one promising solution to enhance the performance is by letting the UAVs learn the environment by various sensors and adapt their movement and communications resource allocation in real time. Thus, the implementation of intelligent learning algorithms are common in designing UAV-networks for various purposes including navigation, deployment and anti-jamming.

Despite of the benefits in enabling UAV-BSs, there are many remaining issues to be addressed. A significant one is to find suitable UAV-BSs' positions when deploying the UAV-BSs network. Since the life time of the battery powering one UAV-BS is limited and the number of available UAV-BSs is also constrained, UAV-BSs should be deployed in an energy-efficient method. Another critical challenge is the design of the movement strategy for UAVs. Since in realistic situations, in order to take advantage of the high mobility of UAVs, it is important that a reasonable strategy needs to be designed for UAVs to cope with various environments.

1.2 Air to Ground (A2G) Channel Model

The A2G channel adpoted follows that in [1] where line-of-sight (LoS) occurs with a certain probability. The probability of a LoS and non line-of-sight (NLoS) channel between UAV j at horizontal position $m_j = (x_j, y_j)$ and user i at horizontal location $u_i = (\tilde{x}_i, \tilde{y}_i)$ are formulated as [1]

$$\begin{aligned} P_{LoS} =& \frac{1}{1 + a\exp(-b(\frac{180}{\pi}\tan^{-1}(\frac{H_j}{r_{ij}}) - a))}, \\ P_{NLoS} =& 1 - P_{LoS}, \end{aligned} \tag{1.1}$$

where H_j is the altitude of UAV-BS j; a and b are environment dependent variables; $r_{ij} = \sqrt{(x_j - \tilde{x}_i)^2 + (y_j - \tilde{y}_i)^2}$ is the horizontal euclidean distance between the i^{th} user and j^{th} UAV. Then the path loss for LoS and NLoS can be written as

$$\begin{aligned} PL_{LoS} &= 20\log(\frac{4\pi f_c d_{ij}}{c}) + \eta_{LoS}, \\ PL_{NLoS} &= 20\log(\frac{4\pi f_c d_{ij}}{c}) + \eta_{NLoS} \end{aligned} \tag{1.2}$$

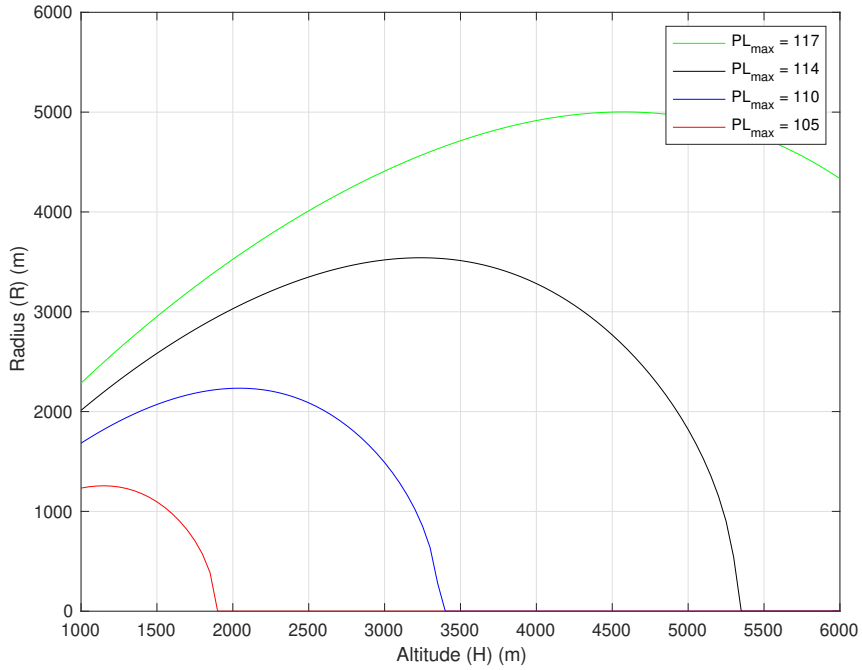

Figure 1.1: Radius vs. altitude curve for different maximum path loss.

where f_c is the carrier frequency, c is the speed of light and d_{ij} denotes the distance between between the UE and UAV-BS given by $d_{ij} = \sqrt{H_j^2 + r_{ij}^2}$. Moreover, η_{LoS} and η_{NLoS} are the environment dependent average additional path loss for LoS and NLoS condition respectively. According to (1.1), (1.2), the path loss (PL) can written as:

$$\begin{aligned} PL =& PL_{LoS} \times P_{LoS} + PL_{NLoS} \times P_{NLoS} \\ =& \frac{A}{1 + a\exp(-b(\frac{180}{\pi}(\frac{H}{r}) - a))} + 20\log\frac{r}{\cos(\frac{H}{r})} + B \end{aligned} \tag{1.3}$$

where $A = \eta_{LoS} - \eta_{NLoS}$ and $B = 20\log(\frac{4\pi f_c}{c}) + \eta_{NLoS}$.

In order to show the effect of different PL_{max} on the radius-altitude curve, we have plotted this relation (2.2) in Fig. 1.1 where the coverage radius is a function of both, the altitude H and the PL_{max}, by keeping a constant environment parameters such as those of urban.

The path loss for UEs which are associated with the GBSs at distance r_{ik} can be modeled by $PL_{ik} = \eta_B r_{ik}^{\alpha_B}$ where η_B is the additional PL over the free space PL and α_B is the PL exponent.

Moreover, the signal-to-interference-plus-noise ratio (SINR) experienced at a UE

at a distance r_{ij}/r_{ik} from its associated UAV-BS j or GBS k can be expressed as

$$SINR_{ij/ik} = \frac{P_{j/k} h_0 PL^{-1}_{ij/ik}}{\sigma^2 + \sum_{\bar{j} \in Q \setminus j, \bar{k} \in O \setminus k} I_{i\bar{j}/i\bar{k}}}, \tag{1.4}$$

where

$$I_{i\bar{j}/i\bar{k}} = P_i h_0 PL^{-1}_{i\bar{j}/i\bar{k}}, \tag{1.5}$$

represents interference from other UAV-BSs/GBSs, $P_{j/k}$ represents the transmit power of the serving base station. h_0 is the small fading gain assumed to be an independent number following the exponential distribution and σ^2 is the variance of the additive white Gaussian noise component. Therefore, according to the Shannon Capacity Theorem, the data rate C_i of of the i^{th} UE can be expressed as $C_i = B \log_2(1+SINR_{ij/ik})$ where B is the bandwidth of the channel.

1.3 Genetic Algorithm

Genetic Algorithm (GA) works on a population which consists of some candidate solutions and the population size is the total number of solutions. Each solution is considered to be a chromosome and each chromosome has a set of genes where each gene is represented by the features of the solutions. Then, each individual chromosome has a fitness value which is computed based on the fitness function representing the quality of the chromosome. Moreover, a selection method called roulette wheel method where the chromosome with higher fitness value has a higher chance to survive the population.

However, the selection process can only generate the best candidate solution with no more change of the chromosome. In order to ensure the diversity of the solution to avoid falling into local optimal solutions, crossover and selection are applied after selection process. In crossover procedure, two chromosome are selected in a probability of crossover rate to exchange information so new chromosomes are generated. Also, in mutation procedure, each chromosome has a probability of mutation rate to replace a set of genes with new random values. This process repeats for t iteration until t reach a preset iteration limit. Fig. 1.2 illustrates the general work flow of a complete GA.

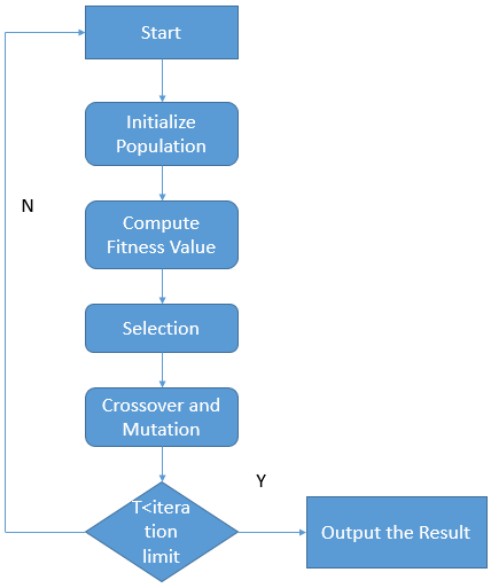

Figure 1.2: GA workflow

1.4 Related Work

Research on UAV-BSs development has focused on finding horizontal positioning [10]-[12] and altitude optimization [13]-[15]. In [9] and [11], an identical coverage radius is assumed for all UAV-BSs. The work in [9] proposes an efficient spiral placement algorithm aiming to minimize the required number of UAVs, while [11] models the UAV deployment problem based on circle packing theory and study the relationship between the number of deployed UAV-BSs and the coverage duration. In [6], the authors use a K-Means clustering method to partition the ground users to k subsets and users belonging to the same subset are served by one UAV. All these works have a fixed altitude assumption. The relationship between the altitude of UAV-BSs and the coverage area is studied in [1] and [5]. In [1], the method of finding the optimal altitude of a single UAV placement for maximizing the coverage is studied based on a channel model with probabilistic path loss (PL). Reference [5] formulates an equivalent problem based on the same channel model as [1] and proposes an efficient solution. Moreover, [3] studies multiple UAV-BS 3D placements with a given radius taking into account energy efficiency by decoupling the UAV-BS placement in the vertical dimension from the horizontal dimension. In recent year, artificial intelligence algorithms are growing in various research fields. The authors in [2] applied GA which is a popular artificial intelligence algorithm to derive the optimal UAV locations in

5G applications with the consideration of energy consumption and coverage range.

Moreover, machine learning techniques have begun to gain popularity to be utilized in deploying UAVs [7]-[9]. In particular, in [16], a machine learning framework based on Gaussian mixture model (GMM) and a weighted expectation maximization (WEM) algorithm to predict the locations of UAVs in which the total power consumption is minimized are proposed. Also, the authors in [4] study a Q-learning based algorithm to find the optimal trajectory to maximize the sum rates of ground users for a single UAV base station (UAV-BS). Reference [8] proposes a deep reinforcement learning based movement design for multiple UAV-BSs.

1.5 Report Outline

The structure of the report is as followed:

Chapter 2 presents an introduction of reinforcement learning.

Chapter 3 presents a Optimal 3D Deployment method for deploying UAV Base Stations.

Chapter 4 describes a reinforcement learning based method to obtain the UAV movement strategy in UAV-assisted network.

Chapter 5 makes concluding remarks and discusses future work.

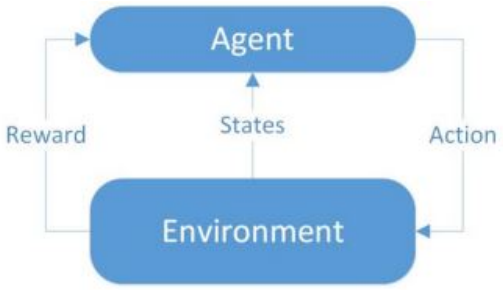

Figure 2.1: RL workflow

Chapter 2

Reinforcement Learning

Fig. 2.1 illustrates the workflow of a basic reinforcement learning (RL) method. The RL task is to train an agent who interacts with the environment that provides feedback to each of its actions. The agent arrives at different states by performing actions. Actions lead to rewards so we reinforce the agents to learn to choose the best actions based on the rewards. Therefore, the only objective of the agent is to maximize its total reward across an episode. The way the agent chooses its actions is known as policy. The RL examples include Q-learning, deep Q-learning, policy gradient and etc.

2.1 Neural Network

The materials presented in this section follow [10]

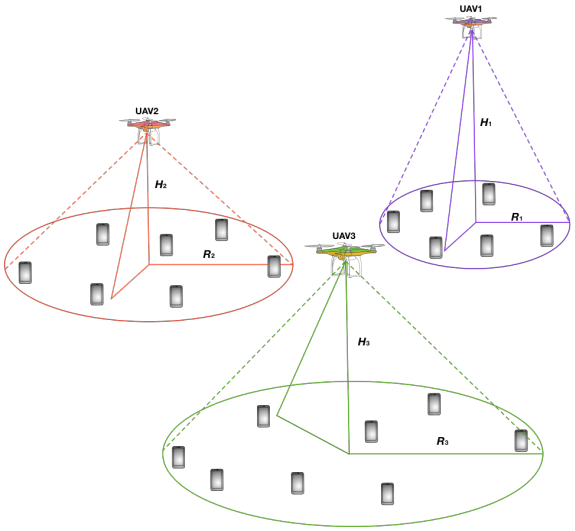

Figure 2.2: A communication system model of multiple UAV-BSs serving ground users

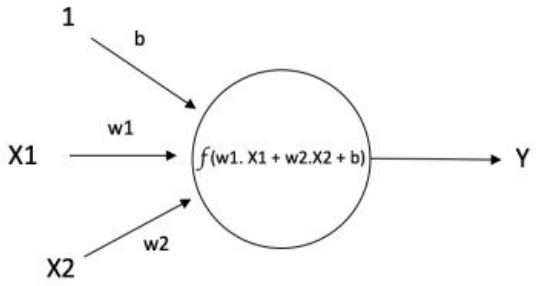

Figure 2.3: A single neuron

2.1.1 A Single Neuron

The basic unit of a neural network is called neuron which receives numerical input from some other nodes, or from an external source and computes an output. Each input has an associated weights and a bias and the nerual applies an activation function to the weighed sum of the inputs as shown in Fig. 2.3. The purpose to have an activation function is to have a non-linear representation of the outputs. In neural network, the sigmoid function is used as activation function.

$$f(x) = sigmoid(x) = \frac{1}{1 + \exp(-x)} \tag{2.1}$$

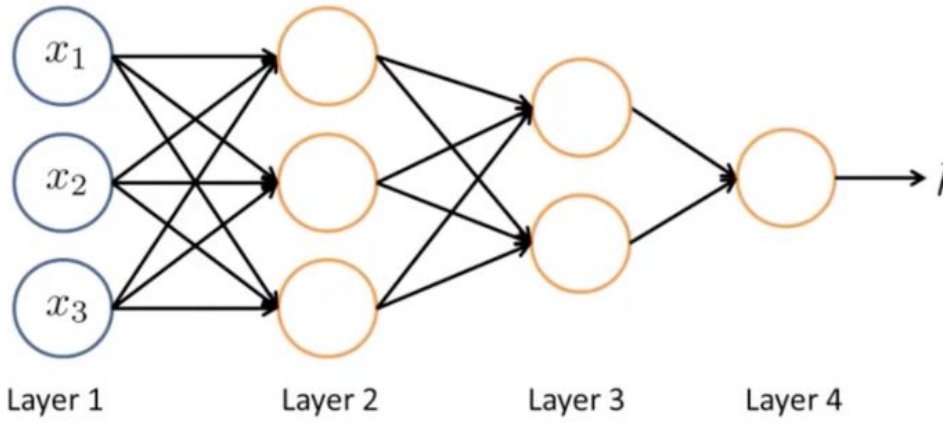

Figure 2.4: Fully Connected Neural Network

2.1.2 Feedforward Neural Network

A feedforward neural network is a collection of neurons which are connected with each other in a particular way. A neuron takes inputs from other neurons and output the computation results to another neuron. Fig. 2.4 shows a simple fully connected neural network. The layer1 is called input layer and layer 4 is called output layer. The layers in between are called hidden layer. The neurons in one layer are connected with all the neurons from previous layer. In a feedforward neural network, the information moves in only one direction which is forward. It goes through the neurons in hidden layers and to the neurons in output layers without any loop.

2.1.3 Backpropagation

Initially all the weights in the neurons are randomly assigned. For the inputs from the training dataset, the neural network takes those inputs and the outputs can be derived. The outputs are compared with the desired outputs so that the difference between the computed outputs and desired outputs can be observed. According to the difference which is also known as propagated, the values of weights can be adjusted until the propagated is below a predetermined threshold.

Once the above algorithm terminates, we consider the nerual network is ready to take inputs which are not from training dataset to accurately predict the outputs.

2.1.4 Q-Learning

Q-learning is an off policy reinforcement learning algorithm which finds the best action for a given state. It's considered off-policy because the q-learning function learns from actions that are outside the current policy. More specifically, q-learning learns a policy that maximizes the total reward.

- Q-Value: The Q-Value $Q(s, a)$ represents the total rewards of agents being at state s and performing action a and the Q-Value for each state and action can be found in the Q-Table. It can be computed by equation:

$$Q(s, a) = r(s, a) + \gamma max_a Q(s', a) \tag{2.2}$$

  The above equation states that the Q-value which is derived from the agent being at state s and taking action a equals to the immediate reward $r(s, a)$ plus the highest possible Q-value of the next state s'. γ is the discount factor which represents the contribution of future rewards.

- Q-Table: Q-Table is a look up table which states the Q-Value that represents the future values for actions for each states. Fig. 2.5 illustrates the format of a Q-Table where the Q-Value for actions to each states are stated.

To begin with, the Q-Table is initialized with all zeros. Then the agent chooses an action based on epsilon greedy strategy that 90% the agent chooses the action with highest Q-Value while 10% the agent chooses a random action. After, based on the action the agent chooses, the reward of performing the action is observed. According to the outcome and the reward, Q-Value can be updated based on equation:

$$Q_{new}(s, a) = Q_{old}(s, a) + \alpha(r(s, a) + \gamma maxQ(s', a) - Q_{old}(s, a)) \tag{2.3}$$

2.1.5 Deep Q-Network

The traditional Q-Learning is a powerful algorithm to create a look up table for the agent so that the agent is capable for making rational action in each state. However, the drawback of Q-Learning is when there are too many states in the environment, it requires a large amount of memory since we need a long Q-Table. Therefore, the

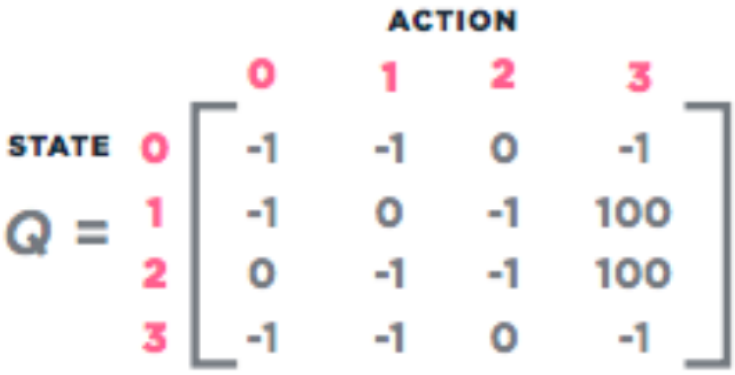

Figure 2.5: A simple Q-Table

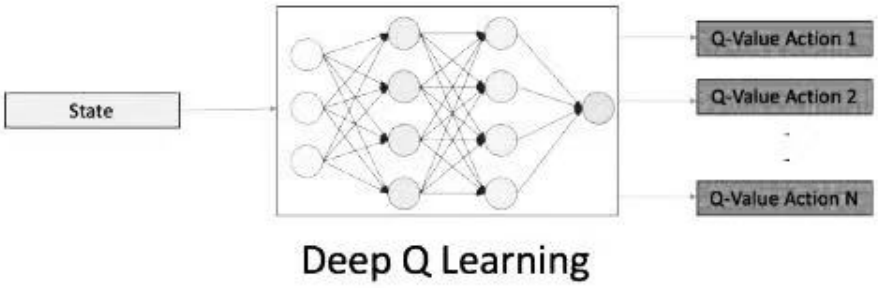

Figure 2.6: Deep Q-Network

neural network is a powerful tool that can be utilized to estimate Q-value as shown in Fig. 2.6.

Therefore, the next action is determined by the maximum output of the neural network. Refer to equation (2.3), if we make the loss function $Loss = (r + \gamma max_a \tilde{Q}(s', a; \Theta) - Q(s, a; \Theta))^2$ where Θ represents the parameters of the Q-Network, it becomes a simple regression problem.

However, in this loss function, $Q(s, a; \Theta)$ plays the role of a desired target in a regression problem which needs to be stationary in order to converge the network. Therefore a separate network is used to calculate the target. This target network has the same architecture as the the network to predict Q-Value but with frozen parameters. The parameters of predicted network are copied to target network in every C iterations and C is a predetermined value.

Also, another important factor in Deep Q-Network is experience replay. It stores a fixed size of samples from training data into a memory tuple. In each training step, a mini-batch of samples are randomly selected from the memory to train the Q-Network. Experience replay breaks up the correlation in the training data by

sampling batch of experiences randomly from a large memory pool which also helps the network to converge.

Chapter 3

QoS-Compliant Optimal 3D Deployment Method

The 3D deployment of UAV-BSs can be decomposed into the 2D horizontal locations optimization and altitude determination. This is because the UAV altitude only impacts the cell radius and path loss experienced in the cell, while the horizontal location and a radius determine which UEs are covered by the UAV. As clearly seen in Fig. 1.1, for a given PL_{max}, there is a maximum radius R_{max} and a corresponding altitude H_{max}. If the altitude is smaller or larger than H_{max}, while maintaining the same radius, the path loss on the cell edge will be larger than the given PL_{max}. Since the cell radius affects the total number of the covered UEs, we want the cell radius to be maximized in order to potentially cover more users. Hence the 3D deployment solution takes the procedure as follows. First, a maximum cell radius upper bound R_{max} that guarantees the desired PL_{max} requirement is derived. Second, the 2D placements of $|Q|$ UAVs and their respective coverage radii bounded by R_{max} that maximize the total number of UEs supported while satisfying the individual data rate requirements and the UAV capacity constraint are formulated and solved. Finally, given the actual coverage radius of each UAV obtained from the second step, the altitude that leads to the achieved minimum cell edge path loss is determined.

3.1 System Model

Fig. 4.1 shows a communication network model where many UEs are clustered to be served by multiple UAV-BSs. The objective is to find the optimal locations for

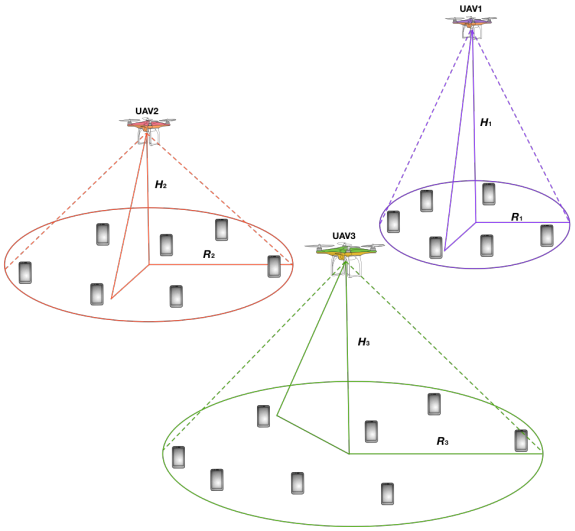

Figure 3.1: A communication system model of multiple UAV-BSs serving ground users

UAV-BSs so that the ground users' coverage ratio and the coverage radii can be maximized. Let $\mathcal{P}$ be the set of all the UEs which are labelled as $i = 1, 2, ... |\mathcal{P}|$. Each UE has a unique data rate requirement c_i and all UEs have a maximum tolerated path loss PL_{max}that serves the purpose to guarantee all the data rate requirements from UEs are feasible, for QoS compliance. $\mathcal{Q}$ denotes the set of available UAV-BSs labelled as $j = 1, 2, ... |\mathcal{Q}|$ and each UAV-BS has a data rate capacity C_j. In our system, we assume that no ground base station is available but the locations and data rate requirement of all users are known. Despite of the known interference issue in UAV cells, this work does not take into account multi-cell interference, which may be mitigated by various techniques such as beamforming, frequency planning, etc.

3.2 Problem Formulation of Finding Optimal 3D Location of UAV-BS

3.2.1 2D UAV-BS Deployment Problem

Since we model the 2D deployment problem via placing multiple circles of different sizes, unlike authors in [2] who investigate a problem of solving for the least number of UAVs to cover users in a region, this problem is equivalent to finding the appropriate location and radius for each UAV-BS to cover as many UEs as possible while simultaneously satisfying the data rate requirements and the UAV capacity constraint.

A binary variable $\gamma_{ij} \in \{0, 1\}$ is used to indicate whether or not the user i is covered by UAV-BS j, 1 for service and 0 for no coverage. The necessary condition for user i to be covered by UAV-BS j is that the horizontal Euclidean distance between them is less than the coverage radius of UAV-BS j, R_j, which can be written as $\|m_j - \gamma_{ij} u_i\| \leq R_j$. Following [11], the constraint equation can be rewritten as $\|m_j - \gamma_{ij} u_i\| \leq R_j + B(1 - \gamma_{ij})$, where M is a large constant which is larger than the largest horizontal distance between a user and a UAV so the constraint holds in any conditions. If a user is within the serving area of a UAV-BS, the UAV-BS can allocate certain data channels to the user which has a unique data rate requirement c_i. For simplicity, we assume that for any UE, the allocated data rate equals what it requires. Then the data rate allocation problem can be expressed as $\sum_{j=1}^{|\mathcal{Q}|} c_i \gamma_{ij} \leq C_j$, where C_j is the data capacity of UAV j.

Now, the deployment problem becomes a rucksack-like problem which is a NP-hard problem. It can be expressed as

$$
\begin{aligned}
&\underset{R_j, m_j}{\text{maximize}} \sum_{j=1}^{|\mathcal{Q}|} \sum_{i=1}^{|\mathcal{P}|} \gamma_{ij}, \\
\text{s.t. } & C1 : \|m_j - \gamma_{ij} u_i\| \leq R_j + M(1 - \gamma_{ij}), \\
& \quad i \in \{1, 2, \ldots |\mathcal{P}|\}, j \in \{1, 2, \ldots |\mathcal{Q}|\}, \gamma_{ij} \in \{0, 1\} \\
& C2 : \sum_{i=1}^{|\mathcal{P}|} c_i \gamma_{ij} \leq C_j, j \in \{1, 2, \ldots |\mathcal{Q}|\} \\
& C3 : \sum_{j=1}^{|\mathcal{Q}|} \gamma_{ij} \leq 1, i \in \{1, 2, .. |\mathcal{P}|\} \\
& C4 : R_j \leq R_{max}, j \in \{1, 2, \ldots |\mathcal{Q}|\}
\end{aligned}
\tag{3.1}
$$

Our objective is to maximize the number of served users. First, $C1$ in (5), guarantees that a UE can be served by a UAV-BS, when the horizontal distance between the UE and the UAV-BS is less than UAV-BS's coverage radius. Then $C2$ regulates that the total data rate of all covered users served by one UAV-BS cannot exceed the data rate capacity of the UAV-BS. Furthermore, $C3$ ensures each user should be served by at most one UAV-BS. Last, Fig. 1.1 shows that the function of coverage radius respective to altitude for a given PL_{max} is a concave function so there exists a maximum radius R_{max} that any coverage radii $R > R_{max}$ does not have a feasible solution. Thus, $C4$ ensures that the radii of UAV-BSs are no larger than R_{max}. A genetic algorithm to

solve this optimization problem will be presented in the next section.

3.2.2 Finding the Optimal Altitude for UAV-BS

After, the horizontal locations and coverage radii of UAV-BSs have been determined and all the coverage radii are less than R_{max}. Therefore, for each UAV-BS, the range of altitude which results in the PL value less than PL_{max} can be obtained from Fig. 1.1. The objective for this step is to find the optimal altitude for each UAV-BS which requires least transmit energy, ie., the minimum path loss, to provide service for the coverage range derived in step 1.

As observed from (2.2), the path loss between a UAV-BS and UE is a function of the horizontal distance r and the altitude H, that is, $PL = f(r, H)$. Also, from Fig. 1.1, for a given PL_{max}, defining the elevation angle $\theta = \frac{H}{R}$, there exists an elevation angle θ_{max} that maximizes the radius R by solving $\frac{\partial R}{H} = 0$. As derived in [1], θ_{max} satisfies the following equation:

$$\frac{\pi}{9\ln(10)}\tan(\theta_{max}) + \frac{abA\exp(-b(\frac{180}{\pi}\theta_{max} - a))}{(a\exp(-b(\frac{180}{\pi}\theta_{max} - a)) + 1)^2} = 0 \tag{3.2}$$

where θ_{max} is environment dependent so it is a constant in a given environment. It has been proven by [3] that this elevation angle provides the minimum PL of the users in the boundary which is equivalent to the PL of all the UEs within the covered range are minimized so the required transmit power of the UAV-BS is minimized. Therefore, once the actual coverage radius R of each UAV-BS is obtained in Subsection III-A, the UAV-BS altitude H_{opt} is given by $H_{opt} = R\tan(\theta_{max})$. Fig. 3.2 shows the relationship between PL and altitude for given radii. It can be observed that as long as the radius is fixed, a minimum value of PL always exists.

3.3 GA based UAV-BS Deployment Strategy

As illustrated in Algorithm 2, the horizontal location, and the coverage radius of each UAV-BS are treated as a gene in the GA model. Therefore, for UAV-BS j, the combination (x_j, y_j, R_j) is a gene. Placing genes for all the available UAV-BSs together, i.e., $\{x_j, y_j, R_j\}_{j\in\mathcal{Q}}$ makes a chromosome. The required inputs include K, D, $\mathcal{P}$, $\mathcal{Q}$, R_{max}, $\{c_i\}_{i\in\mathcal{P}}$, $\{u_i\}_{i\in\mathcal{P}}$, θ_{opt}, p_m, p_c where K is the number of iterations for finding the optimal result, D, p_m and p_c are the population size, mutation rate and

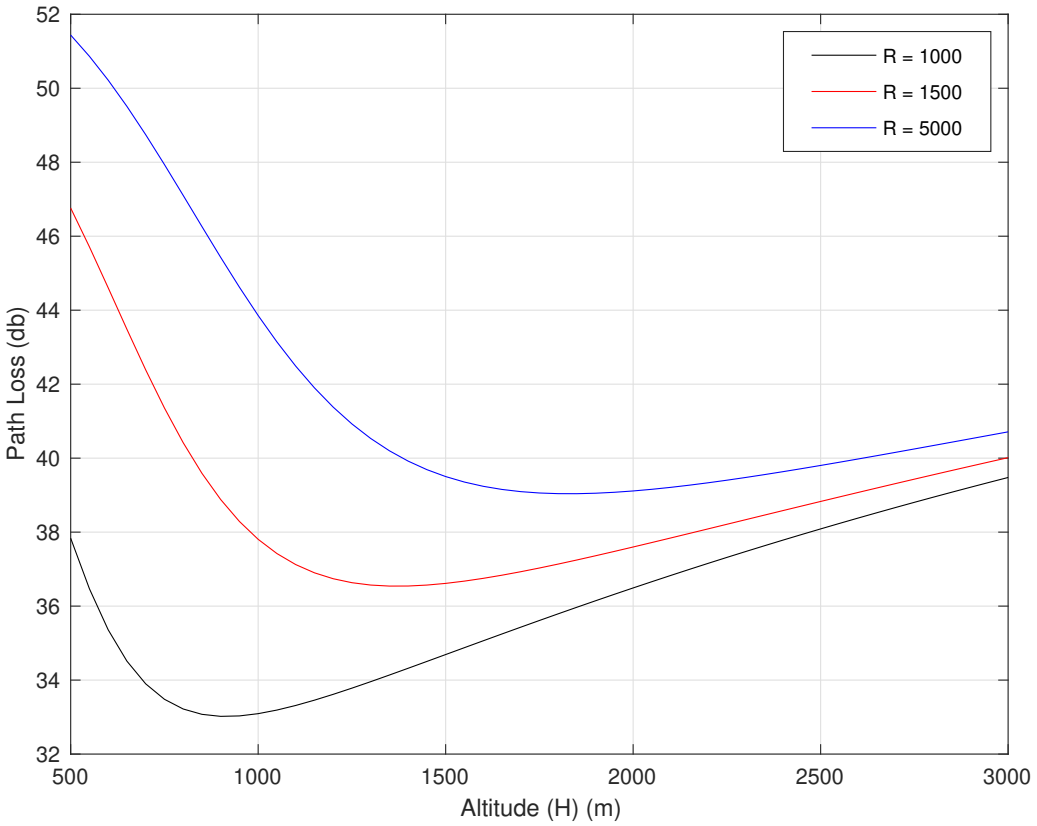

Figure 3.2: Path loss vs. altitude for given radii in urban environment.

crossover rate for GA respectively. The outputs are the horizontal locations, altitudes and coverage radii, denoted by $O_j, j = 1, 2, ... |\mathcal{Q}|$, of all the UAV-BSs.

First, $|\mathcal{Q}|$ empty lists are created and each of them is to store the covered UEs of the corresponding UAV-BSs. Also, two arrays r, $\hat{r}$ are created, respectively, to store the number of covered UEs in each UAV-BS and the total number of covered UEs of all UAV-BSs known as the fitness score. In step 3, the first population ν_1 is generated by creating D chromosomes where the horizontal locations of all UAV-BSs are initialized by assigning each of them with the equidistant point of 3 random UEs' locations, and the coverage radius are initialized by generating random numbers in the range from 1 to R_{max}.

Then, K iterations are executed to find the 2D deployment result from Step 4 to Step 20. In Step 5 and Step 6, if the horizontal distance between a UE and a UAV-BS is less than the coverage radius, the UE can be served by the UAV-BS. Also, if a UE is within the coverage range of more than one UAV-BS, it is assigned to

the closest one. In the for loop from Step 7 to Step 16, calculate the sum data rate $\sum_{\hat{p}\in\mathcal{O}_j} c_{\hat{p}}$ of all covered UEs for each UAV-BS. If the sum data rate is smaller than the data capacity C_j, the number of covered UEs $|\mathcal{O}_j|$ is stored to array r. Otherwise, a negative number is stored to array $\hat{r}$ and the algorithm breaks out the loop and goes back to Step 5, which means the fitness of this chromosome is negative. In Step 15, the fitness function of the chromosome is the total number of covered UEs and it is saved into array $\hat{r}$.

In Step 17, the roulette wheel method is applied to update the current population $\nu_{\hat{k}}$. A random chromosome is selected within the current population to be the competitor. Comparing the fitness score of all the chromosomes with the competitor, the chromosomes with less fitness scores are replaced by the competitor. Afterward, in the crossover procedure, p_c of chromosomes are randomly selected and paired. Each pair is considered to be the parent chromosomes. In each parent chromosomes, the first half genes of one chromosome and second half genes of the other chromosome are exchanged to produce children chromosomes. In Step 19, all the chromosomes have a probability of p_m to perform mutation process in which one gene of the mutated chromosome is selected to be replaced by random horizontal location and coverage radius.

Finally, in Step 21 and Step 22, we can obtain the result of horizontal locations and coverage radii of UAV-BSs via choosing the chromosome with the maximum fitness score. Finally, the optimal altitudes are obtained by $H_{opt} = R\tan(\theta_{max})$.

3.4 Numerical Result

In our simulations, we consider the UEs are uniformly distributed in a 5000 m $\times$ 5000 m area. Referring to [1], the environment parameters are set up as followed: f_c = 2 GHz, PL_{max} = 110 dB, (a, b, η_{LoS}, η_{NLoS}) is configured to be (4.88, 0.43, 0.1, 21), (9.61, 0.43, 0.1, 20), (12.08, 0.11, 1.6, 23), (27.23, 0.08, 2.3, 34) corresponding to suburban, urban, dense urban and high-rise urban environments, respectively. The GA parameters set (K, D, p_m, p_c) is configured to be (10000, 100, 0.01, 0.8). Also, we assume there are three different data rate requirements of all UEs, $c^1 = 5 \times 10^6$ bps, $c^2 = 2 \times 10^6$ bps and $c^3 = 1 \times 10^6$ bps, and each UE has one of these three data rate requirements. Moreover, all the UAV-BSs have the same data rate capacity $C = 1 \times 10^8$ bps. Fig. 3.3 illustrates the UE distribution and the GA deployment result with 100% coverage percentage.

Algorithm 1 Genetic Algorithm Based 3D UAV-BS Placement Method

Input: K, D, $\mathcal{P}$, $\mathcal{Q}$, R_{max}, $\{c_i\}_{i\in\mathcal{P}}$, $\{u_i\}_{i\in\mathcal{P}}$, θ_{opt}, p_m, p_c
Output: $\{m_j\}_{j\in\mathcal{Q}}$, $\{H_j\}_{j\in\mathcal{Q}}$, $\{R_j\}_{j\in\mathcal{Q}}$

1: Create $|\mathcal{Q}|$ empty list $\{O_j\}_{j\in\mathcal{Q}}$.
2: Create two arrays r, $\hat{r}$ with size $|\mathcal{Q}|$ and D respectively.
3: **Initialize Population:** Initialize first population ν_1 by creating D sets of chromosomes.
4: **for** $\hat{k}=1$; $\hat{k}\leq K$; $\hat{k}++$ **do**
5: **for** $\hat{i}=1$; $\hat{i}\leq D$; $\hat{i}++$ **do**
6: Perform UE assignments according to their distances to UAV-BSs and store the results in $\{O_j\}_{j\in\mathcal{Q}}$.
7: **for** $j=1$; $j\leq|\mathcal{Q}|$; $j++$ **do**
8: **if** $\sum_{\hat{p}\in O_j} c_{\hat{p}} \leq C_j$ **then**
9: $r\,[j] \leftarrow |O_j|$
10: **else**
11: $\hat{r}[\hat{i}] \leftarrow -100$
12: Continue and go back to step 5
13: **end if**
14: **end for**
15: **Fitness Function:** $\hat{r}[\hat{i}] = sum(r)$
16: **end for**
17: **Selection:** update $\nu_{\hat{k}}$ using roulette wheel method to select chromosomes
18: **Crossover:** Based on p_c, update $\nu_{\hat{k}}$ by exchanging information of parent chromosomes to produce children chromosomes
19: **Mutation:** Based on p_m, gene are selected randomly to be replaced new random values
20: **end for**
21: Find the chromosome with maximum value in $\hat{r}$ and obtain $\{m_j\}_{j\in\mathcal{Q}}$ and $\{R_j\}_{j\in\mathcal{Q}}$ from the chromosome.
22: Obtain $\{H_j\}_{j\in\mathcal{Q}}$ by solving $H_{opt} = R\tan(\theta_{opt})$
23: **return** $\{H_j\}_{j\in\mathcal{Q}}$, $\{R_j\}_{j\in\mathcal{Q}}$, $\{m_j\}_{j\in\mathcal{Q}}$

Fig. 3.4 shows the average coverage ratios of 80 UEs by 8 available UAV-BSs with 10 realizations in four different environments when increasing the number of UAV-BSs. The UEs are arbitrarily distributed. As seen from Fig. 3.4, the coverage ratio varies significantly in four deployment scenarios, particularly with high-rise urban one much more challenging than others.

By applying Shannon Capacity Theorem, the required SNR of each UAV-BS can be calculated through $C = B\log_2(1+\frac{P_r}{P_n})$, where B is the bandwidth of the channel, P_r and P_n denote the required received power and average noise power, respectively. In our model, we assume that $B = 1\times10^7$ Hz, $P_r = -74$ dBm and $P_n = -100$ dBm. Thus, we can obtain the minimum required power for each UAV-BS by $P_t = P_r + PL(R_j, H_j)$. Fig. 3.5 shows that in urban environments with 10 available UAV-BSs and 3 different approaches to determine altitudes, the average minimum required transmit power of all UAV-BSs when increasing the number of UEs. In

the fixed altitude approach, all the UAV-BSs are deployed in the same altitude. In random altitude approach, each UAV-BS is deployed into a random altitudes. The altitudes from both fixed altitude and random altitudes are selected from the range where PL_{max} requirement is met. As we can see, if the UAV-BSs are deployed in the altitude in the way we proposed, less average transmit power is required to provide wireless service.

For further performance comparison, we test 3 algorithms to obtain the coverage percentage of UEs given 10 available UAV-BSs fixing environment parameters (Urban). In each test, we generated 10 times of arbitrary UE distributions of 80, 200 and 450 UEs respectively. Besides the GA deployment strategy proposed, we have two other schemes for comparison. The first one is random placement which randomly selects a location within the square region and a coverage radius. The second one is K-means algorithm which partitions the UEs into $\hat{K}$ clusters to be covered by $\hat{K}$ UAV-BSs. The results are presented in TABLE 4.1. Compared with these two other algorithms, GA has the significant advantage of solving the optimization problem with many variables involved. It is observed that the result of GA based deployment has higher coverage percentage and this advantage is more pronounced when the number of UEs increases.

Table 3.1: Coverage ratio comparison in urban environment.

$\lvert\mathcal{P}\rvert$	80	200	450
GA Deployment Method	99.2%	88.6%	75.3%
K-Means	98.6%	82.3%	69.4%
Random	85.6%	72.1%	59.4%

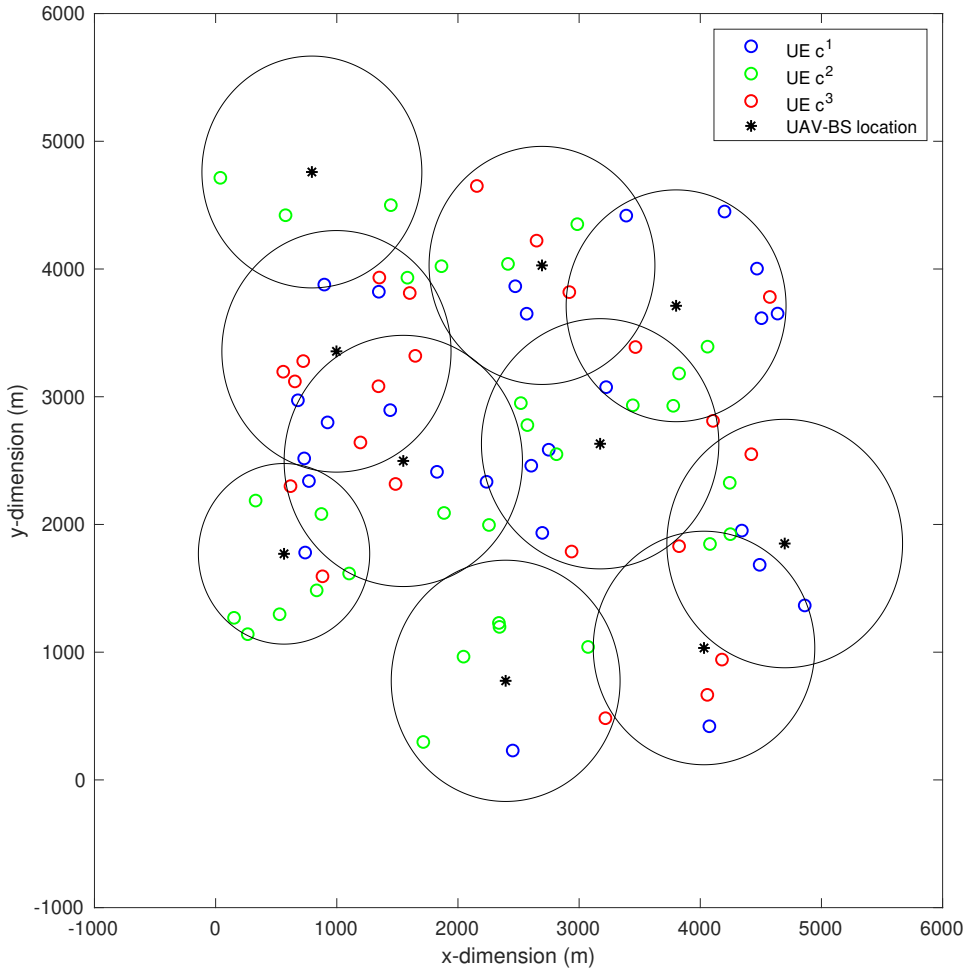

Figure 3.3: The 100% coverage ratio result of GA deployment with 80 UEs in a 5000 m $\times$ 5000 m square region with different data rate requirements.

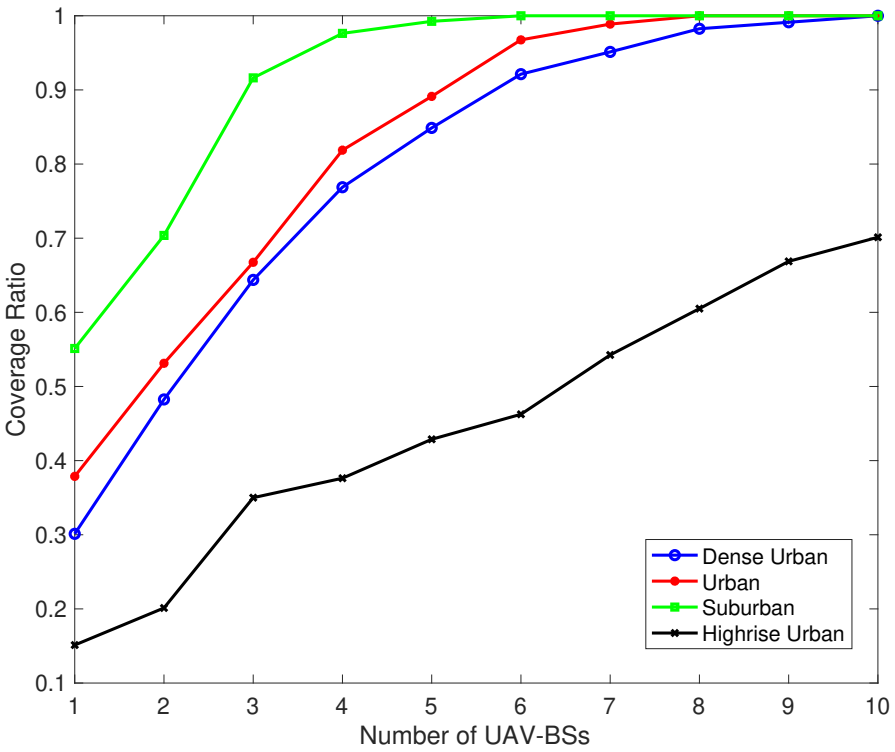

Figure 3.4: The coverage ratio versus the number of UAV-BSs in four environments.

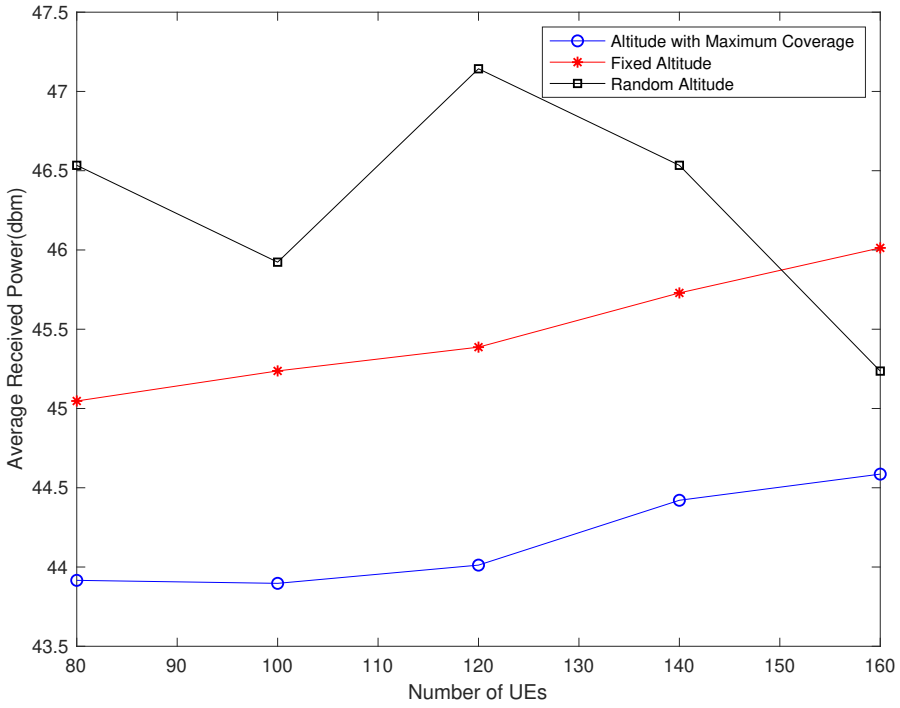

Figure 3.5: The UAV's average transmit power comparison of altitude with maximum coverage, fixed altitude and random altitude in urban environments.

Chapter 4

Dynamic Movement Strategy in a UAV-Assisted Network

In this chapter, we investigate a real-time dynamic UAV movement strategy design on a deep reinforcement learning framework called Deep Q-Network (DQN) [14] to maximize the sum data rate. Our contribution lies in the formulating the design problem of the UAVs' movement strategy to find the optimal locations of UAVs in every single time instant, in response to the ground users' movement.

4.1 UAV-Assisted Network System Description

Fig. 4.1 shows the framework of UAV-assisted wireless communications system model where UAVs serve as aerial base stations and provide wireless communications to the ground UEs. Also, the traditional terrestrial infrastructures are capable of serving the UEs which are not covered by UAV-BSs. Let $\mathcal{P}$ be the set of all the UEs which are labelled as $i = 1, 2, ..., |\mathcal{P}|$. $\mathcal{Q}$ denotes the set of available UAV-BSs labelled as $j = 1, 2, ..., |\mathcal{Q}|$ and $\mathcal{O}$ denotes the set of ground base stations (GBSs) labelled as $k = 1, 2, ..., |\mathcal{O}|$. In our system, we assume that the UEs are assigned to the closest base station to receive wireless communication service and all the UAV-BSs cells are deployed at the same altitude H. Also, considering the existing interference mitigation technologies, both the interference between UAV-BSs and interference between UAV-BSs and GBSs cells are assumed to be negligible. Moreover, ground users are assumed to move at each time instant so the locations of each UE at each time instant can be expressed as $m_i(t) = [x_i(t), y_i(t)], t \in T$ where T is the time window considered..

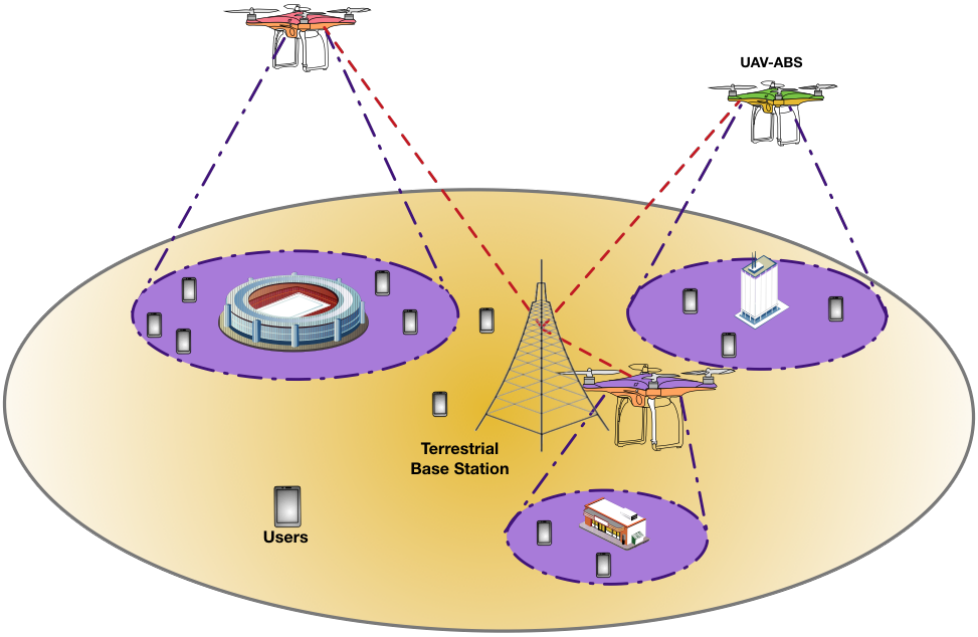

Figure 4.1: A communication system model of UAV-assisted Network

Similarly, the locations of UAV-BS j can be written as $n_j(t) = [\tilde{x}_j(t), \tilde{y}_j(t)]$. Also, $u_k = [\check{x}_k, \check{y}_k]$ denotes the the location of the k-th GBS, which is a known parameter in the study.

4.2 UAV Dynamic Movement Problem Formulation

The dynamic UAV-BS movement strategy problem can be treated as a design of determining the positions of the UAV-BSs at each time instant. The objective is to find the optimal positions for all UAV-BSs in each time-slot, to maximize the sum data rates of users. $\gamma_{ij/ik}(t)$ is a binary variable indicating whether the user i is associated with UAV-BS j or GBS k at time instant t, with 1 for service and 0 for no association. Thus, the optimization problem at each time instant t can be formulated as:

$$
\begin{aligned}
&\underset{n_j(t), j\in\mathcal{Q}}{\text{maximize}} \sum_{i=1}^{|\mathcal{P}|} C_i(t), \\
\text{s.t. } & C1: \|n_j(t) - \gamma_{ij}(t) m_i(t)\| \leq \left\|n_{\bar{j}}(t) - m_i(t)\right\| \\
& \quad + M\left|1 - \gamma_{ij}(t)\right|, \forall j \in Q, \forall \bar{j} \in \{\mathcal{O}, \mathcal{Q} \setminus j\} \\
& C2: \|u_k - \alpha_{ik}(t) m_i(t)\| \leq \|u_{\bar{k}} - m_i(t)\| \\
& \quad + M\left|1 - \alpha_{ik}(t)\right|, \forall k \in \mathcal{O}, \forall \bar{k} \in \{\mathcal{Q}, \mathcal{O} \setminus k\} \\
& C3: \sum_j \gamma_{ij}(t) + \sum_k \alpha_{ik}(t) = 1, \forall i, j, k.
\end{aligned}
\tag{4.1}
$$

Constraints $C1$ and $C2$ in (6) guarante all the UEs are associated with the nearest UAV-BSs/GBSs. Also, M is a large number to ensure the constraints hold in any UE association conditions. Then $C3$ guarantees all the UEs are associated with only single base station. Therefore, the objective of the optimization problem is to find the optimal positions of UAV-BSs in each instant over time duration T so that the sum data rates of the users can be maximized.

4.3 Deep Q-Network based UAV Movement Strategy

In this section, given the real-time locations of a set of UEs, we present a reinforcement learning based UAV-BS movement strategy to obtain the optimal real-time locations of UAV-BSs. Before discussing the movement of UAV-BSs, the mobility model of UEs needs to be discussed first. The random walk model [13] is chosen as the UE mobility model in this letter, but other models can be easily included. The moving direction of UEs are uniformly distributed among left, right, forward, backward and staying still. Moreover, the initial positions of the ground users are assumed to be fixed. At each instant $t \in T$ when ground users move, all UAV-BSs take action in response to the movement of the ground users.

The objective is to train a neural network to represent the action-value function which takes the local observations of the positions of both UEs and UAV-BSs in any instant as inputs and derives the action-value functions of the UAV-BSs movement. The Deep Q-Network consists of four parts: states, actions, rewards and the Q-Network. At each time slot t, each agent observes a state s_t, from the state space S

Algorithm 2 Deep Q-Network Based UAV-BS Movement Strategy

Required: Initial Position of UAV-BSs, $m_i(0)$

Initialize replay memory D with capacity N, initialize action-value network $\bar{Q}$ with weight $\bar{\Theta}_{j\in Q}$, target network $\tilde{Q}$ with weight $\tilde{\Theta}_{j\in Q}$ with random weights.
for each episode **do**
  Reset UAV-BSs to the initial positions
  for each time step t **do**
    for each UAV-BS agent j **do**
      Observe $s_t^{(j)}$
      Choose the action a_t^j which maximizes the $\bar{Q}(s_t^j, a_t^j; \bar{\Theta}_j)$
    end for
    All agents take actions, observe rewards r_t^j, update state $s_t^j \rightarrow s_{t+1}^j$
    for each UAV-BS agent j **do**
      Observe s_{t+1}^j
      Store (s_t^j,a_t^j, r_{t+1}^j, s_{t+1}^j) into replay memory D_j
      Uniformly sample mini batch from replay memory D_j
      Perform a gradient descent on $Loss = (r_t^j + \beta max_{a'} \tilde{Q}(s_{t+1}^j, a'; \tilde{\Theta}_j) - \bar{Q}(s_t^j, a_t^j; \bar{\Theta}_j))^2$ with respect to network parameters $\bar{\Theta}_j$.
      Update $\tilde{\Theta}_j = \bar{\Theta}_j$ every C time steps
    end for
  end for
end for

and takes an action a_t in the action space A based on the decision from Q-Network $\bar{Q}$. The principle of the Q-Network is to obtain the maximum Q-value which maximizes the sum data rates of UEs. Following the action, the state of each agent transits to a new state s_{t+1} and the agents receive a reward r_t which is determined by the instantaneous sum data rates of ground users.

4.3.1 State Representation

All agents' states are defined as: $s = (x_{uav}, y_{uav})$ which is the horizontal position of the UAVs. Assuming that the initial states of all UAV-BSs are at the optimal positions where the sum data rates of ground users are maximized at time instant t_0. The optimal positions can be derived by conducting exhaust search.

4.3.2 Action Space

At each time step, all the UAV-BSs take an action $a_t \in A$ which includes choosing a direction for UAV-BSs to move according to the current state s_t, based on the decision from Q-Network $\bar{Q}$. In our model, we assume that all UAV-BSs move in the

same speed in any time step therefore the moving distances for any UAV-BSs from any time instant t to $t+1$ are assumed to be the same. More specifically, since we assume that all the UAV-BSs are at the same altitude H, there are 5 different actions in A: (1,0) means the UAV-BS will turn right, (-1,0) means the UAV-BS will turn left, (0,1) means the UAV-BS will move forward, (0,-1) means the UAV-BS will move backward and (0,0) means the UAV-BS will stay still.

4.3.3 Reward Design

After performing an action, the UAV-BS has a different location so the UEs need to change the association based on problem **??**. Therefore, the new association comes with a new instantaneous sum data rates of the ground UEs. The principle of designing the reward function is to improve the UEs' instantaneous data rates, which enables the agent to receive a positive reward. When the action results in a reduction of the sum data rates of the UEs, the UAV-BS receives a negative reward. Thus, the reward function can be expressed as

$$r_t = \begin{cases} 1, & \text{if sum rates increase} \\ -0.2, & \text{if sum rates remain the same} \\ -1, & \text{if sum rates decrease} \end{cases} \tag{4.2}$$

4.3.4 Training Procedure

The training procedure requires a learning rate α and a discount factor β. The learning procedure is divided into several episodes, and at the beginning of each episode, the positions of UAV-BSs will be reset to the initial values. We leverage a DQN with experience replay to train the agents [14]. Each agent j has a DQN $\bar{Q}$ that takes an input of the observation of the current state s_t^j and generate the output of the value functions corresponding to all the actions. At each training step t, each agent chooses the action a_t^j which leads to the maximum estimated Q value. Based on the action taken by the agent, the transition tuple $(s_t^j, a_t^j, r_t^j, s_{t+1}^j)$ is collected and stored into the replay memory D with a size of N. Then, in each episode, a predetermined size of mini-batch experiences E are uniformly sampled to update Θ using gradient

descent method to minimize the loss function

$$Loss = \sum_{E}(r_t^j + \beta max_{a'}\tilde{Q}(s_{t+1}^j, a'; \tilde{\Theta}_j) - \bar{Q}(s_t^j, a_t^j; \bar{\Theta}_j))^2 \tag{4.3}$$

where $\tilde{\Theta}_j$ is the parameter set of a target network $\tilde{Q}$ which is replaced by the parameter set $\bar{\Theta}_j$ of training Q-Network $\bar{Q}$ every C time steps. The experience replay can improve the training efficiency by breaking the correlation between samples so as to stabilize the training.

4.4 Numerical Result

In our simulation, we consider UAV-assisted model in a 5000 m $\times$ 5000 m area and uniformly divide the area into 4 sections, that is, Section 1 : $0 < x \leq 2500, 0 < y \leq 2500$, Section 2 : $2500 < x \leq 5000, 0 < y \leq 2500$, Section 3 : $0 < x \leq 2500, 2500 < y \leq 5000$, Section 4 : $2500 < x \leq 5000, 2500 < y \leq 5000$. We assume that initially all of the UEs are distributed in the whole area, and then in the middle of the time duration, the majority (90%) of the UEs converge to Section 1. At the end of the time duration, all the UEs go back to the uniformly distributed in the whole area. The UEs follow random walk mobility model inside the section area. There is one GBS available located at $u_0 = [2500, 2500]$. Moreover, referring to [1], the environment parameters are set up as follows: $f_c = 2$ GHz, $PL_{max} = 103$ dB, (a, b, η_{LoS}, η_{NLoS}) is configured to be (9.61, 0.43, 0.1, 20) corresponding to the urban environment. The transmit powers of UAV-BSs and GBS are set to be 37 dBm and 40 dBm, respectively. Also, the Deep Q-Network parameter set (α, β, N, B, C) is configured to be (0.01, 0.9, 2000, 50, 200). Fig. 4.2 shows the UEs distribution and their association in a time instant. The UEs and base stations with same color represent the association and all the UEs are associated with the closest base stations.

Table 4.1: Comparisons of processing time of different algorithm

NA	Processing Time (ms)
Deep Q-Network	210
Exhaust Search	4117
K-Means	387
Fixed	0

Fig. 4.3 shows the comparison of the sum data rates in all the time instants with different algorithms. It can be observed that the overall performance in 500

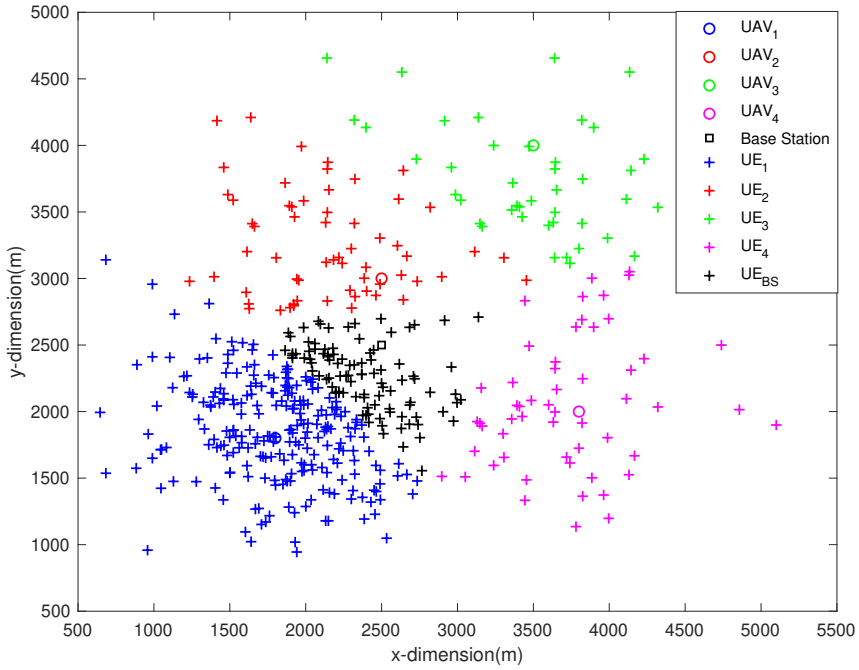

Figure 4.2: The UE distribution and association with 500 UEs in a 5000 m × 5000 m area

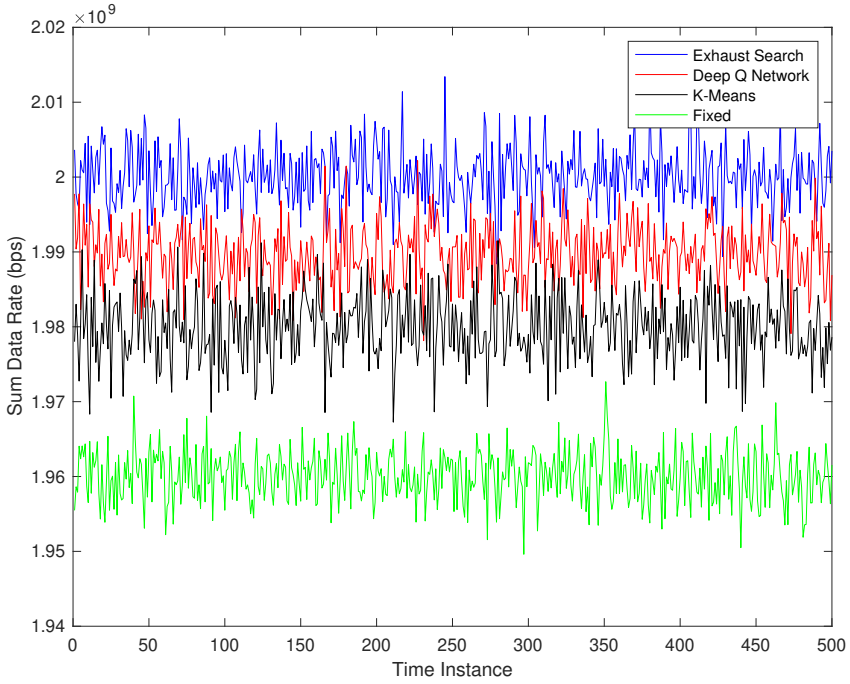

Figure 4.3: sum data rate comparison of different methods.

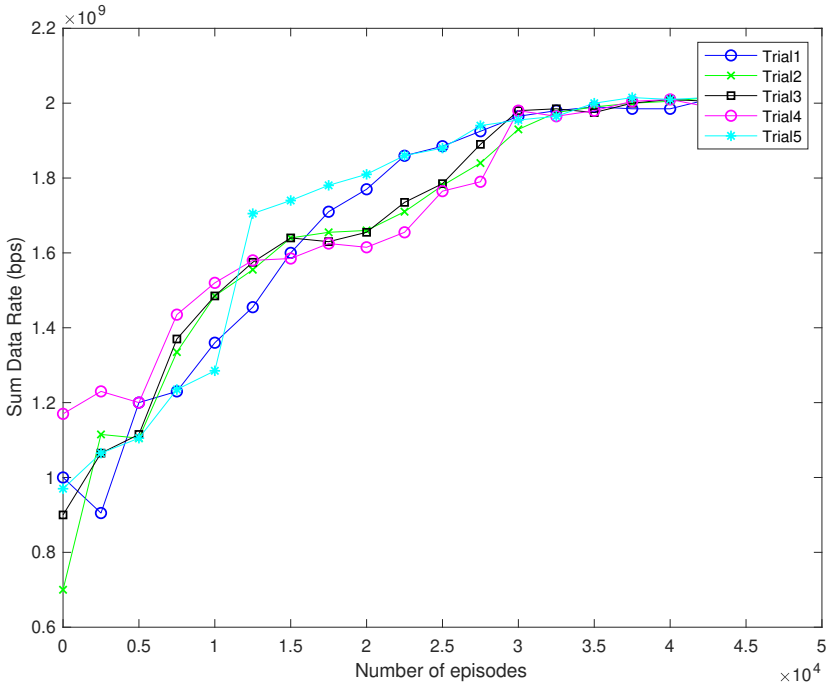

Figure 4.4: Sum data rate versus number of training episodes.

time instant of Deep Q-Network outperforms the fixed location or K-Means deployment strategy and closely follows the performance of the Exhaust Search. However, considering the computation cost analysis in Table 4.1, which is generated via using Intel® Core™ i5- 4430 Processor to run the algorithm 10 times and take the average processing time, exhaustive search as expected achieves the highest performance but the computation complexity can be too high for real-time processing. The Deep Q-Network performs close to the exhaustive search but with significantly less processing resource and time, which is particularly critical for low-latency communications and mission execution involving UAVs.

Fig. 4.4 further plots the sum data rates against the number of training episodes. It can be observed that the UAV-BSs are capable of carrying out their actions via iterative learning from their past experience to improve the performance.

Chapter 5

Conclusion and future work

5.1 Optimal 3D Location of UAV-BS with Maximum Coverage

In this report, we have proposed and evaluated a cost-efficient 3D UAV-BS deployment algorithm for providing real-life wireless communication services when all the UEs are randomly distributed with various data rate requirements. A GA-based deployment algorithm has been proposed to maximize the number of covered UEs, and simultaneously meet the UEs' individual data rate requirements and UAV-BS capacity limit. The proposed algorithm outperforms other algorithms in requiring less UAV-BSs to provide full coverage of all UEs.

5.2 Optimal UAV Dynamic Movement Strategy

In this report, we have also proposed and evaluated a dynamic UAV-BS deployment algorithm for optimizing the real-time performance of wireless communication services when all the UEs are moving. A Deep Q-Network based algorithm has been proposed to maximize the sum data rates of ground UEs in a dynamic UAV-assisted network. Results show that the proposed algorithm outperforms other existing dynamic deployment algorithms.

5.3 Future Work

There are several issues appear to be worthwhile for future research. The first one is that our work is based on an assumption that the interference between UAVs can be negligible by various inteference management technologies. However, in the practical implementation of UAV network, this is an essential factor which can not be ignored. Thus, in our future research, the interference between UAVs will be taken into consideration. Moreover, another critical factor of UAV network is the energy consumption. A more effective energy model of UAV network is another research field with great potential.

Appendix A

Genetic Algorithm Python Implementation

```
# Fitness function
def evaluate(chromosome, UEpoint, uav_number,
    data_rate_capacity, data_requirement):

    score = 0
    UAV_assig = distribute_UE(chromosome, UEpoint,
        uav_number)
    UAV_assig_array = np.asarray(UAV_assig)
    #print(UAV_assig_array)
    for i in range(uav_number):
        UAV_index = np.where(UAV_assig_array==i)[0]
        #print(UAV_index.shape)
        sum_data_rate = 0
        for item in UAV_index:
            sum_data_rate += data_requirement[item]
        if sum_data_rate > data_rate_capacity:
            print("Data rate excceds capacity")
            score = -1
            break
        else:
            score += len(UAV_index)
```

```
    return score


# Initial population
def getIntialPopulation(k, populationSize,
    maximum_radius):
    """

    :param k: UAV number
    :param populationSize: population number
    :param data_in: UE_location
    :return:
    """
    chromosomes = np.zeros((populationSize, k*3),
        dtype=np.float)
    uav_position_final = np.zeros((populationSize, k
        * 2), dtype=np.float)
    #UEPoints = np.random.randint(100,
        size=(UE_number, 2))
    for i in range(populationSize):
        uav_position = np.zeros((k, 3),
            dtype=np.float)
        uav_location = np.zeros((k, 2),
            dtype=np.float)
        for j in range(k):
            random = np.random.rand(2)*5000
            random_radius = np.random.rand()*1300
            uav_position[j,0] = random[0]
            uav_position[j,1] = random[1]
            uav_position[j,2] = random_radius
            uav_location[j, 0] = random[0]
            uav_location[j, 1] = random[1]
        chromosomes[i,:] = uav_position.flatten()
```

```
            uav_position_final[i,:] =
                uav_location.flatten()

        return chromosomes, uav_position_final

    # crossover to generate new descendant
    def crossover(population, pc, uav_number):
        """
        :param population: chromosomes
        :param pc: probability of crossover is 0.8
        :return: new population
        """
        # The number of the chromosomes need to be
            considered based on pc
        m, n = population.shape
        numbers = np.uint8(m * pc)
        # Make sure the number is even
        if numbers % 2 != 0:
            numbers += 1
        # Generate an empty structure
        updatepopulation = np.zeros((m, n),
            dtype=np.float)
        # Generate random index
        index = rd.sample(range(m), numbers)
        # Copy the unused ones
        for i in range(m):
            if not index.__contains__(i):
                updatepopulation[i, :] = population[i, :]
        # Crossover
        while len(index) > 0:
            a = index.pop()
            b = index.pop()
            # Generate a cross over point
            crossoverPoint_index =
                np.random.randint(uav_number, size=1)
```

```
            crossoverPoint_index = 3*crossoverPoint_index
            crossoverPoint = crossoverPoint_index[0]
            # one-single-point crossover
            updatepopulation[a, 0:crossoverPoint] =
                population[a, 0:crossoverPoint]
            updatepopulation[a, crossoverPoint:] =
                population[b, crossoverPoint:]
            updatepopulation[b, 0:crossoverPoint] =
                population[b, 0:crossoverPoint]
            updatepopulation[b, crossoverPoint:] =
                population[a, crossoverPoint:]
        return updatepopulation



def select(chromosomes, fit_,
    tournament_size):#Tournament selection
        """

        :param tournament_size: tournament size
        :param chromosomes: chromosomes
        :param fit_: fitness result
        :return:
        """
        m, n = chromosomes.shape
        newpopulation = np.zeros((m, n), dtype=np.float)
        # Check validity of tournament size.
        if tournament_size >= m:
            msg = 'Tournament size ({}) is larger than
                population size ({})'
            raise ValueError(msg.format(tournament_size,
                m))
        # Select a father and a mother.
        for i in range(m):
```

```
            competitors = rd.sample(range(m),
                tournament_size)
            newpopulation[i, :] =
                chromosomes[max(competitors, key=lambda x:
                fit_[x]), :]
        return newpopulation


def mutation(chromosomes, pm, uav_number):
    """
```

Appendix B

Deep Q-Network Python Implementation

```
import numpy as np

    def _build_net(self):
        self.s = tf.placeholder(tf.float32, [None,
            self.n_features])
        self.q_target = tf.placeholder(tf.float32,
            [None, self.n_actions])
        with tf.variable_scope('Q_net'):
            c_names, n_l1, w_initializer,
                b_initializer = \
                ['Q_net_params',
                    tf.GraphKeys.GLOBAL_VARIABLES], 10,
                    \
                tf.random_normal_initializer(0.,
                    0.3), tf.constant_initializer(0.1)

            with tf.variable_scope('l1'):
                w1 = tf.get_variable('w1',
                    [self.n_features, n_l1],
                    initializer=w_initializer)
```

```
            b1 = tf.get_variable('b1', [1, n_l1],
                initializer=b_initializer)
            l1 = tf.nn.relu(tf.matmul(self.s, w1)
                + b1)

        # second layer. collections is used later
            when assign to target net
        with tf.variable_scope('l2'):
            w2 = tf.get_variable('w2', [n_l1,
                self.n_actions],
                initializer=w_initializer,
                collections=c_names)
            b2 = tf.get_variable('b2', [1,
                self.n_actions],
                initializer=b_initializer,
                collections=c_names)
            self.q_eval = tf.matmul(l1, w2) + b2

    with tf.variable_scope('loss'):
        self.loss =
            tf.reduce_mean(tf.squared_difference(self.q_target,
            self.q_eval))
    with tf.variable_scope('train'):
        self._train_op =
            tf.train.RMSPropOptimizer(self.lr).minimize(self.loss)

    self.s_ = tf.placeholder(tf.float32, [None,
        self.n_features])    # input
    with tf.variable_scope('target_net'):
        c_names = ['target_net_params', tf]

        with tf.variable_scope('l1'):
            w1 = tf.get_variable('w1',
                [self.n_features, n_l1],
                initializer=w_initializer)
```

```
                b1 = tf.get_variable('b1', [1, n_l1],
                    initializer=b_initializer,
                    collections=c_names)
                l1 = tf.nn.relu(tf.matmul(self.s_,
                    w1) + b1)


            with tf.variable_scope('l2'):
                w2 = tf.get_variable('w2', [n_l1,
                    self.n_actions],
                    initializer=w_initializer)
                b2 = tf.get_variable('b2', [1,
                    self.n_actions],
                    initializer=b_initializer)
                self.q_next = tf.matmul(l1, w2) + b2

    def store_transition(self, s, a, r, s_):

        transition = np.hstack((s, [a, r], s_))

        index = self.memory_counter % self.memory_size
        self.memory[index, :] = transition

        self.memory_counter += 1

    def choose_action(self, observation):
        observation = observation[np.newaxis, :]

        if np.random.uniform() < 0.1:
            actions_value =
                self.sess.run(self.q_eval,
                feed_dict={self.s: observation})
            action_chosen = np.argmax(actions_value)
        else:
```

```
        action_chosen = np.random.randint(0,
            self.n_actions)
    return action_chosen
```

Bibliography

[1] A. Al-Hourani, S. Kandeepan, and S. Lardner. Optimal lap altitude for maximum coverage. *IEEE Wireless Communications Letters*, 3(6):569–572, December 2014.

[2] F. Al-Turjman, J. P. Lemayian, S. Alturjman, and L. Mostarda. Enhanced deployment strategy for the 5g drone-bs using artificial intelligence. *IEEE Access*, 7:75999–76008, 2019.

[3] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu. 3-D placement of an unmanned aerial vehicle base station (UAV-bs) for energy-efficient maximal coverage. *IEEE Wireless Communications Letters*, 6(4):434–437, August 2017.

[4] H. Bayerlein, P. De Kerret, and D. Gesbert. Trajectory optimization for autonomous flying base station via reinforcement learning. In *Proc. IEEE 19th Int. Workshop Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5, June 2018.

[5] 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 *Proc. IEEE Int. Conf. Communications (ICC)*, pages 1–5, May 2016.

[6] B. Galkin, J. Kibilda, and L. A. DaSilva. Deployment of UAV-mounted access points according to spatial user locations in two-tier cellular networks. In *Proc. Wireless Days (WD)*, pages 1–6, March 2016.

[7] Yiming Huo, Xiaodai Dong, Tao Lu, Wei Xu, and Marvin Yuen. Distributed and multi-layer uav networks for next-generation wireless communication and power transfer: A feasibility study. *IEEE Internet of Things Journal*, 2019.

[8] X. Liu, Y. Liu, and Y. Chen. Reinforcement learning in multiple-UAV networks: Deployment and movement design. *IEEE Transactions on Vehicular Technology*, 68(8):8036–8049, August 2019.

[9] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim. Placement optimization of UAV-mounted mobile base stations. *IEEE Communications Letters*, 21(3):604–607, March 2017.

[10] Donald Michie, David J Spiegelhalter, CC Taylor, et al. Machine learning. *Neural and Statistical Classification*, 13, 1994.

[11] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage. *IEEE Communications Letters*, 20(8):1647–1650, August 2016.

[12] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Mobile unmanned aerial vehicles (uavs) for energy-efficient internet of things communications. *IEEE Transactions on Wireless Communications*, 16(11):7574–7589, November 2017.

[13] J. Ren, G. Zhang, and D. Li. Multicast capacity for vanets with directional antenna and delay constraint under random walk mobility model. *IEEE Access*, 5:3958–3970, 2017.

[14] Mnih Volodymyr, Kavukcuoglu Koray, Silver David, A Rusu Andrei, and Veness Joel. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[15] Y. Zeng, R. Zhang, and T. J. Lim. Wireless communications with unmanned aerial vehicles: opportunities and challenges. *IEEE Communications Magazine*, 54(5):36–42, May 2016.

[16] Q. Zhang, M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Machine learning for predictive on-demand deployment of uavs for wireless communications. In *Proc. IEEE Global Communications Conf. (GLOBECOM)*, pages 1–6, December 2018.