Performance Enhancement of Aerial Base Stations via Reinforcement Learning-based 3D Placement Techniques

by

Nahid Parvaresh

A thesis
presented to the University of Ottawa
in fulfillment of the
thesis requirement for the degree of
Master of Computer Science Concentration in Applied Artificial Intelligence

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Deploying unmanned aerial vehicles (UAVs) as aerial base stations (BSs) in order to assist terrestrial connectivity, has drawn significant attention in recent years. UAV-BSs can take over quickly as service providers during natural disasters and many other emergency situations when ground BSs fail in an unanticipated manner. UAV-BSs can also provide cost-effective Internet connection to users who are out of infrastructure. UAV-BSs benefit from their mobility nature that enables them to change their 3D locations if the demand of ground users changes. In order to effectively make use of the mobility of UAV-BSs in a dynamic network and maximize the performance, the 3D location of UAV-BSs should be continuously optimized. However, solving the location optimization problem of UAV-BSs is NP-hard with no optimal solution in polynomial time for which near optimal solutions have to be exploited. Besides the conventional solutions, i.e. heuristic solutions, machine learning (ML), specifically reinforcement learning (RL), has emerged as a promising solution for tackling the positioning problem of UAV-BSs. The common practice for optimizing the 3D location of UAV-BSs using RL algorithms is assuming fixed location of ground users (i.e., UEs) in addition to defining discrete and limited action space for the agent of RL.

In this thesis, we focus on improving the location optimization of UAV-BSs in two ways: 1-Taking into account the mobility of users in the design of RL algorithm, 2-Extending the action space of RL to a continuous action space so that the UAV-BS agent can flexibly change its location by any distance (limited by the maximum speed of UAV-BS).

Three types of RL algorithms, i.e. Q-learning (QL), deep Q-learning (DQL) and actor-critic deep Q-learning (ACDQL) have been employed in this thesis to step-by-step improve the performance results of a UAV-assisted cellular network. QL is the first type of RL algorithm we use for the autonomous movement of the UAV-BS in the presence of mobile users. As a solution to the limitations of QL, we next propose a DQL-based strategy for the location optimization of the UAV-BS which largely improves the performance results of the network compared to the QL-based model. Third, we propose an ACDQL-based solution for autonomously moving the UAV-BS in a continuous action space wherein the performance results significantly outperforms both QL and DQL strategies.

Acknowledgements

First, I would like to express my deepest appreciation to Dr. Hasan Ural who believed in me as a passionate student who wanted to continue her studies in uOttawa. Without him, I would not have started this journey. Although he is no longer with us, his soul will live forever in my heart.

Next, I would like to offer my sincere gratitude to my inspiring supervisor, Dr. Burak Kantarci, for his continuous supports, kindness, great knowledge and insightful comments throughout the completion of my Master's. I was so lucky to have got an opportunity to work under his supervision in Smart Connected Vehicle Innovation Centre. The environment he provides to his students, the friendly atmosphere, and the amazing projects, kept me always alive and motivated throughout this journey.

Finally, I would like to thank my dear husband, Mehdi Razmjoo, for always encouraging me to do better, supporting me emotionally, making me laugh when I was bored and exhausted, staying awake with me for projects and exams and all the sacrifices he made. This challenging journey would have been so tough without him.

Table of Contents

List of Tables vii

List of Figures viii

List of Symbols xii

1 Introduction 1

1.1 Practical use cases for aerial base stations 4

1.2 Challenges in the implementation of aerial base stations 6

1.3 Motivation 8

1.4 Contributions 10

1.5 Research Methodology 11

1.6 Structure of the thesis 11

2 Related works 13

2.1 Methodologies adopting legacy solutions for the optimal placement of UAV-BSs 14

2.2 Methodologies adopting AI-based solutions for optimal placement of UAV-BSs 22

3 Q-learning-based deployment of UAV-BSs 32

3.1 Introduction 32

3.2 System model 33

3.2.1 Channel model . . . 33
3.2.2 Optimization problem formulation . . . 36
3.3 Preliminaries for Q-learning . . . 38
3.3.1 Reinforcement learning . . . 38
3.3.2 Q-learning . . . 40
3.4 The proposed QL-based deployment of UAV-BSs . . . 42
3.5 Simulation Results . . . 45
3.6 Conclusion . . . 50

4 Deep Q-learning-based deployment of UAV-BSs 51
4.1 Introduction . . . 51
4.2 Preliminaries for deep Q-learning . . . 52
4.3 The proposed DQL-based deployment of UAV-BSs . . . 54
4.4 Simulation Results . . . 56
4.5 Conclusion . . . 62

5 Actor-critic deep Q-learning-based deployment of UAV-BSs 63
5.1 Introduction . . . 63
5.2 Preliminaries for actor-critic deep Q-learning . . . 64
5.3 The proposed ACDQL-based deployment algorithm . . . 66
5.4 Simulation Results . . . 72
5.5 Conclusion . . . 78

6 Conclusion 79
6.1 Future Directions . . . 81

List of Tables

1.1 Practical use cases for aerial base stations 5

2.1 An overview of the studies about the 3D placement of UAV-BS 14

2.2 Optimization objectives used by conventional methodologies for the 3D placement of UAV-BS 20

2.3 Optimization objectives and reward functions used by RL-based methodologies for the 3D placement of UAV-BS 29

3.1 Table of notations used in our proposed system model 34

3.2 Environmental parameters of the simulation (based on [90, 91]) 46

4.1 DQN parameters of the simulation in the proposed DQL-based methodology (selected and fine-tuned based on [84]) 58

5.1 DQN parameters of the simulation in the proposed ACDQL-based methodology 73

List of Figures

1.1 General use cases of UAVs 2
1.2 Illustration of the use cases for UAV-BSs 3

2.1 The greedy and motion algorithms proposed in [82] 17
2.2 Voronoi cells with the UAV-BS over their coverage areas 19
2.3 Taxonomy of algorithms used in the literature for the location optimization of UAV-BSs 26

3.1 System model (all users are mobile) 35
3.2 Reinforcement learning 40
3.3 Q-learning 41
3.4 UAV-BS deployment with QL 42
3.5 Network performance comparison under the two baselines and the proposed QL-based UAV-BS placement 48
3.6 Accumulated reward of the agent in the proposed model 49

4.1 Q-table vs DQN 53
4.2 Algorithm flow of Deep Q-learning 54
4.3 UAV-BS deployment with DQL 55
4.4 Network performance comparison under the three baselines and the proposed DQL-based UAV-BS placement 60
4.5 Comparison of the agent reward under the QL-based baseline and the proposed DQL-based UAV-BS placement 61

4.6 The DQN's loss in the proposed DQL-based UAV-BS placement 62

5.1 Actor-critic deep Q-learning 65
5.2 Proposed continuous actor-critic-based UAV-BS's deployment 68
5.3 The flowchart of the proposed ACDQL-based algorithm for the optimal placement of UAV-BS 71
5.4 Network performance comparison under the four baselines and the proposed ACDQL-based UAV-BS placement 75
5.5 Accumulated reward of the agent in the proposed ACDQL-based model . . 76
5.6 Accumulated TD loss of the agent in the proposed ACDQL-based model . 76
5.7 Users sum data rate comparison under the proposed QL, DQL, and ACDQL-based strategies (Test Phase) 77

Abbreviations

$A2G$: air-to-ground

$ACDQL$: actor-critic deep Q-learning

AI: artificial intelligence

COW: cell of wings

DNN: deep neural network

DQL: deep Q-learning

DQN: deep Q-network

DRL: deep reinforcement learning

GPS: global positioning system

$HAPS$: high altitude platform stations

LoS: line-of-sight

MDP: Markov decision process

$MINLP$: mixed integer non-linear programming

ML: machine learning

MOS: quality of experience

$NLoS$: no-line-of-sight

PSO: particle swarm optimization

QL: Q-learning

QoE: quality of experience

QoS: quality of service

RL: reinforcement learning

$SINR$: signal-to-interference-plus-noise ratio

SNR: signal-to-noise ratio

TD: temporal difference

UAV: Uncrewed aerial vehicle

$UAV - BSs$: UAV-mounted base stations

UE: user equipment

List of Symbols

s	state space
a	action space
$r(s,a)$	the immediate reward the agent receives by performing action a in state s
$V(s)$	the value of state s
$Q(s,a)$	the quality of performing action a in state s
γ	the discount factor
$P(s,a,s')$	the probability of moving from state s to state s' by performing action a
$-\mathcal{R}$	significantly low reward used when the agent moves outside the boundaries
T	training time period of RL
N	number of users
(a,b)	environment type variables
θ_i	elevation angle between i-th user and UAV-BS
$P_{i(\mathrm{LoS})}$,$P_{i(\mathrm{NLoS})}$	probability of LoS and NLoS between i-th user and UAV-BS
r_i	horizontal distance between i-th user and UAV-BS
f_c	carrier frequency
c	speed of light
$\delta_{\mathrm{LoS}}, \delta_{\mathrm{NLoS}}$	additional environmental path loss for LoS and NLoS
R_i	downlink data rate of i-th user
p_i^t	transmitted power from the UAV-BS to i-th user
p_i^r	received power from the UAV-BS to i-th user
p_{uav}	total transmission power of UAV-BS
B	total bandwidth of UAV-BS
γ_i	SNR received from UAV-BS to i-th user
σ	noise power
G_i	power gain from UAV-BS to i-th user
g_i^{tx}	transmitting antenna gain from UAV-BS to i-th user
g_i^{rx}	receiving antenna gain from UAV-BS to i-th user
d_0	far field reference distance
α^a	actor's learning rate
α^c	critic's learning rate

Publications of the Candidate During Master's Studies

Publications that are the direct outcomes of the thesis:

- Nahid Parvaresh, Michel Kulhandjian, Hovannes Kulhandjian, Claude D'Amours and Burak Kantarci. "A tutorial on AI-powered 3D deployment of drone base stations: State of the art, applications and challenges." In Elsevier Vehicular Communications, DOI: 10.1016/j.vehcom.2022.100474, 2022. (**Published**)
- Nahid Parvaresh and Burak Kantarci. "Deep Q-learning-enabled deployment of aerial base stations in the presence of mobile users." In the 20th ACM International Symposium on Mobility Management and Wireless Access (MOBIWAC), 2022. (**Published**)
- Nahid Parvaresh and Burak Kantarci. "A continuous actor-critic deep Q-learning-enabled deployment of aerial base stations in the presence of mobile users." In IEEE Internet of Things Journal. (**Under Review**)

Chapter 1

Introduction

Uncrewed aerial vehicles (UAVs), also known as drones[1], are a type of aircraft that can fly without an on-board pilot [87]. UAVs can be autonomously controlled by an on-board unit or they can be controlled manually from the ground [64]. Due to their low cost, high-mobility and hovering capabilities, UAVs are desirable in many civilian applications [31]. For instance, in smart cities, the objective is to provide efficient infrastructure and the provision of continuous services at reduced costs. In such settings, UAVs can play a critical role in delivering various digital services to the users [57]. It is forecast that in the near future millions of drones will be deployed in everyday services to perform a wide range of activities [59]. There are a number of application domains that can be served by UAVs as the last mile service providers. These domains include but are not limited to search and rescue, construction and infrastructure inspection, precision agriculture, delivery of goods, real-time monitoring of road traffic, surveillance, relaying information between long-distance nodes, and providing wireless coverage [72, 56, 42, 48, 28]. Figure 1.1 illustrates the domains in which UAVs can be deployed as the main service provider.

Delivering wireless coverage through UAVs is considered to be one of the most essential factors of future smart cities [16] which is the focus of this thesis. UAV-mounted base stations (UAV-BSs), also called drone-BSs, can provide rapid and cost-effective wireless connectivity for cellular networks [17]. Furthermore, with autonomous decision capability, UAV-BSs can fly autonomously by embedded/built-in microprocessors or distant automated control without human intervention [32].

UAV-BSs are utilized in various contexts to fulfill the connection demands when ter-

[1] As the focus of this thesis is scoped to drone-assisted communications, we hereby use the terms UAV and drone interchangeably.

Figure 1.1: General use cases of UAVs

restrial BSs fall short to provide the connectivity to all users [58]. In case of a temporary ground BS failure, UAV-BSs can quickly hover in the sky in order to prevent users from staying disconnected [30]. In natural disasters and emergency situations where ground BSs are destroyed or overloaded, and rescue teams and citizens do not have access to the Internet, UAV-BSs can be rapidly deployed in the air to provide connectivity [53]. UAV-BSs can also be deployed to locations that have become over-crowded due to an expected event, such as a concert or sport competition, or due to an unexpected event, such as an accident, where the density of connected devices grows above the capacity of the available terrestrial BSs to satisfy their Internet connectivity needs [96, 37]. Furthermore, in rural areas where there is no infrastructure to bring Internet connectivity to users, UAV-BSs can be deployed with low cost to let devices connect to the Internet [96]. Figure 1.2 depicts several use cases when UAV-BSs can provide Internet connectivity to ground users. Indeed, the implementation of terrestrial BSs is a large investment in terms of time and cost which is not always feasible when the demand is urgent, unexpected and/or temporary. In such situations, assisting the cellular networks via UAV-BSs could be a great potential solution. Moreover, UAV-BSs are predicted to be largely exploited in 5G and beyond where a dense network of access points is needed to have ultra reliable low latency connections [34]. In 5G

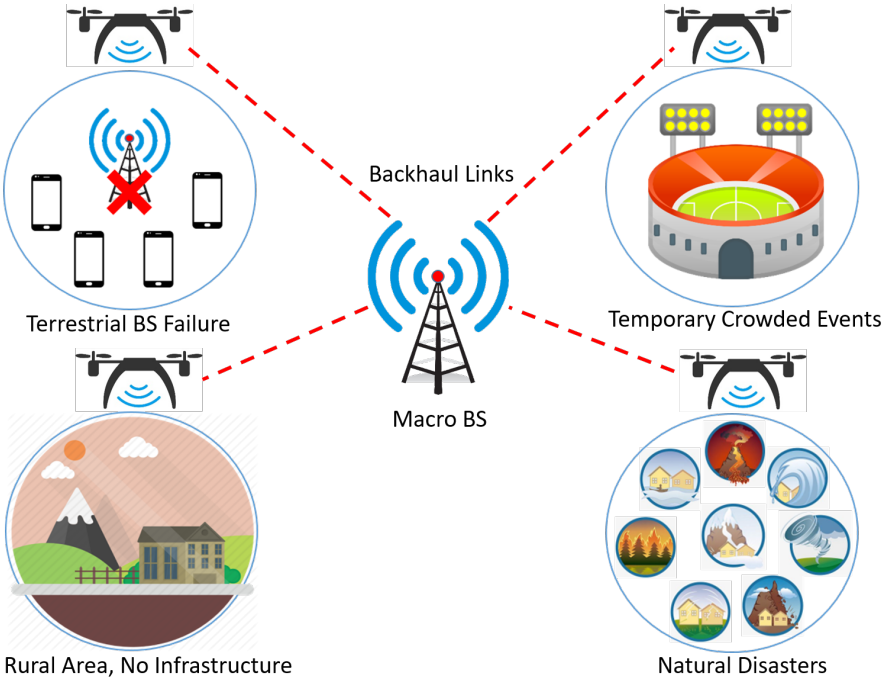


Figure 1.2: Illustration of the use cases for UAV-BSs

wireless networks, picocell access points can also be deployed inside the coverage of macrocells in hotspots of user activity [6]. However, deploying picocells is a time consuming and laborious process that involves training of the personnel and configuration of the access points one by one. An alternative to deploying picocells is the deployment of UAV-BSs in the air in order to complement the existing macrocell infrastructure [34]. According to [23], the deployment of UAV-BSs brings several benefits compared to fixed picocells:

1. **Performance improvement:** Mobility of UAV-BSs enables rapid deployment and positioning around the user equipment (UE) where additional cells are required to meet the demand. Furthermore, since UAV-BSs are placed in high altitude above the earth surface, they are likely to provide line-of-sight (LoS) to UEs which results in higher service quality.

2. **Reduction of deployment cost:** UAV-BSs do not require to be setup like the traditional BSs but only need to be launched to hover over the target area. Service provision while UAV-BSs are hovering is mostly automated and is managed by a central control unit. In light of these, it can be suggested that UAV-BSs are envisioned

to require significantly lower capital expenditures.

3. **Low service overhead:** A typical use case of UAV-BSs is a rapidly occurring excessive demand for Internet connectivity. To cope with rapid demands, an advantage of UAV-BSs over picocells is that, fixed picocells would remain underutilized as hotspots move across different locations. In contrast, UAV-BSs can adapt their position according to the varying demand profiles so to serve users in the new hotspots.

1.1 Practical use cases for aerial base stations

There are several practical usecases where UAV-BSs have been exploited in various projects driven by the industry. In this section, we present a review of four projects that aim at different aspects of UAV-BSs with some overlapping interests.

Facebook's Aquila: Aquila drone, a solar-powered UAV, was developed by Facebook to provide Internet access to the residents in rural areas and/or economically depressed regions. As reported by Facebook, despite the fact that Internet access is an inseparable aspect of urban life, approximately 50 percent of the world population is deprived of connectivity. Indeed, the primary reason for this gap is the high capital and operational costs of laying out and maintaining the infrastructure in those regions, and these roadblocks include but are not limited to land rights, equipment, microwave and fiber. Thus, the traditional business model for connectivity is costly for the operators if they were to bring Internet connectivity to very few potential subscribers in deprived regions. These facts form the basis for Facebook's Aquila project such that the UAV-BSs are deployed to bring affordable Internet connection over a 50 km radius for 90 days to those regions with poor or no connectivity [54]. Although Facebook's Aquila project was halted in 2018 after one of the aircraft's wings was crushed at the time of landing, the concept of bringing connectivity to remote areas via aircrafts has continued [43]. In fact, there are other stakeholders such as Airbus that stepped up to develop high altitude platform stations (HAPS) to reach similar goals as the Aquila project [70].

AT&T's Flying COW: Motivated after the hurricane Maria in Puerto Rico that devastated the northeastern Carribean in 2017 including the collapse of the telecommunication infrastructure, flying cell of wings (COW) is a new technology introduced by AT&T. Flying COW aims at sustaining wireless coverage during natural disasters and big events through drones. A Flying COW drone is designed to deliver wireless coverage to an area of up to 40 square miles in the case of emergency situations [69]. Given a few remaining

cell towers that would be immediately overwhelmed by residents asking for aid, a flying drone is proposed to provide additional network coverage in the aftermath of tornadoes, hurricanes or other disasters to reconnect first responders and residents [66].

SoftBank's Drone Wireless Relay system: SoftBank has been working with Tokyo Institute of Technology since 2019 to conduct research on drone-based wireless relay systems [63]. The drone uses a wired power feed and rises to an altitude of 100 meters to cover an area with a radius of 10km [1]. SoftBank's drone aims to provide connectivity when an outage occurs after natural disasters to save and redirect people towards rescue points. As the SoftBank's drone move in the sky after the disaster, mobile phones can receive signals from a drone, which can determine their locations.

Alphabet and HAPSMobile's HAWK30: HAWK30 is a collaborative project between HAPSMobile and Alphabet's Loon. The goal of the project is to foster the use of solar-powered high altitude drones in the stratosphere in order to deliver connectivity to more regions and more people worldwide. HAWK30 project investigates the use of aerial drones in 5G, which is supposed to transform wireless communications by providing an ultra dense network of access points [80].

Table 1.1 summarizes these use cases by comparing them in terms of objectives.

Table 1.1: Practical use cases for aerial base stations

Project	**Organization**	**Objective**	**Reference**
Aquila	Facebook	Providing Internet access to remote areas with no infrastructure	[54]
Flying COW	AT&T	Providing wireless coverage during natural disasters and big events	[69, 66]
Drone Wireless Relay System	SoftBank	Providing wireless coverage after natural disasters to save people who seek help	[63, 1]
HAWK30	HAPSMobile and Alphabet	Providing an ultra dense network for 5G and beyond	[80]

1.2 Challenges in the implementation of aerial base stations

Despite many benefits and usecases listed in previous sections, due to the air positioning of UAV-BSs, their mobility and autonomous nature and high altitude, UAV-BSs pose several major challenges. Below are some of the key challenges for UAVs as flying BSs:

1. **Channel modeling:** Due to different air section characteristics, air-to-ground/ground-to-air communication differs from ground communications [101]. In particular, any movement or vibration by UAVs might affect the channel characteristics, which could result in poor network performance. Therefore, current communication models which are used for ground-based communications are not applicable for air-to-ground connections [101, 33, 3]. In fact, the UAV-to-ground channels are highly dependent on the type and altitude of UAV, the elevation angle and the propagation environment. Therefore, an accurate channel modeling, specifically designed for air-to-ground communication, is required so to plan, design and evaluate aerial networks effectively.

2. **Optimal 3D placement:** Placing UAV-BSs in optimal locations in the sky (i.e., 3D positioning) and moving them towards the locations with the highest demand is amongst the most challenging concerns about UAV-BSs [98, 60]. Unlike ground BSs that may remain underutilized when UE hotspots change, mobile UAV-BSs can provide rapid connectivity when a need occurs. UAV-BSs are capable of re-positioning themselves in order to move towards the locations where the demand is high. However, the optimization of the best position of UAV-BSs so as to maximize the network performance is an NP-hard problem with no optimal solution in polynomial time [74]. So, non-deterministic algorithms should be used to estimate the near optimal solution which is the focus of this thesis.

3. **Path planning:** Path planning is another challenge for UAV-BSs. As stated in [2, 61], the aim of path planning is to find the optimal, shortest and collision-free path for UAVs to reach their destination. As the energy and flight time of UAVs are limited, an effective path planning algorithm is expected to find the best path for UAVs to serve maximum possible number of users in a limited amount of flight time [12].

4. **Operational altitude:** Higher altitudes of a UAV-BS lead to a higher chance of LoS connectivity for ground users which consequently arises higher service quality [60, 97]. This means the higher altitude of UAVs introduces the higher probability of LoS due to diminishing reflection and shadowing [45]. On the other hand, gradual increase in the altitude of a drone-BS results in a drop in path loss first followed by an increase. The reason of this phenomenon is that the probability of no-line-of-sight (NLoS) in low altitudes is much higher than LoS due to the reflection where the path loss of a NLoS is much higher than LoS [38]. However, the path loss also depends on the distance between the transmitter and receiver [38]. Therefore, beyond a specific altitude, the impact of distance outweighs the other factors and results in higher path loss. Considering that both LoS and path loss have significant impact on the quality of user connectivity, a trade-off between LoS connectivity and path loss needs to be addressed by adjusting the altitude of UAVs, which has been tackled by several studies [53, 100, 73, 92].

5. **Energy limitation:** UAVs operate on rechargeable batteries that limit their flight duration before being recharged [24]. In addition to the conventional energy consumption for communication-related functions, such as communication circuits and signal transmission, UAVs consume excessive power for mobility support [95]. Therefore, advanced charging technologies should be utilized alongside energy efficient protocols and planning in order to prolong the battery lifetime of UAVs so that their operation (flight) time can be extended.

6. **Security and privacy:** As in all other digital systems, security is considered as one of the most crucial concerns for UAV-assisted communications. For UAV-assisted wireless networks, due to the autonomous nature of UAVs and the broadcasting characteristic of the wireless medium, security is even a more momentous problem [46]. According to [93], when launching cyber-attacks, adversaries target the radio links of UAVs which carry information such as data requested by UEs, control signals or global positioning system (GPS) signals that are used for navigation. A compromised UAV-BS poses risks to its subscribers (i.e., served users) since users served by a compromised UAV-BS are likely to lose their cellular connection.

1.3 Motivation

Out of the challenges reviewed above, one of the key challenges for UAV-BSs is the optimal placement of drones which is the focus of this thesis. The position of UAV-BSs directly affects the coverage of the area of interest and how well the quality of service (QoS) requirements of users are met. Also, in 5G and beyond where users' delay is a critical performance indicator of the cellular networks, the service delay experienced by users can be reduced by continuously moving UAV-BSs towards the locations where user density and the Internet demand are higher [104]. Therefore, finding the best possible location of UAV-BSs and how they move in the air in order to improve the performance of network should be precisely investigated. However, finding the optimal 3D placement of UAV-BSs is an NP-hard problem [74, 82], thus, no optimal solution exists in polynomial time [10]. Thus, UAV-BSs should be optimally placed through algorithms that output results at some feasibility levels. In one hand, conventional methods, i.e. heuristic solutions, have been used by various researchers to solve the placement optimization problem of UAV-BSs. But, conventional methods are not suitable for dynamic environments where network topology continuously changes. Conventional algorithms have to rerun repeatedly after even a minimal change occurs in the network [40] in order to adopt with new topology which will result in large computation overhead. Artificial intelligence (AI) and machine learning (ML) techniques on the other hand, can be used in dynamic environments for discovering the best location of UAV-BSs in the air. AI-based methodologies are capable of automatically getting adopted to the minor changes of the environment after a proper training is performed. Amongst all type of ML algorithms, reinforcement learning (RL) and deep reinforcement learning (DRL) are suitable for network-related optimization problems where no data or prior knowledge of the environment is available. The intelligent agent of RL or DRL starts in an unknown environment and takes some actions by trial and error and receives some reward or penalty for each of the actions it performs. The agent gradually learns how to interact with the environment in order to maximize the reward and achieve some specified goals.

There are a number of studies in the literature that have employed RL and DRL, specifically Q-learning (QL) and deep Q-learning (DQL), to solve the placement optimization problem of UAV-BSs. However, these studies have two limitation:

1. The ground users are assumed to be fixed, i.e., they do not move. However, in practical scenarios, users locations change with time that impacts the model performance significantly. Therefore, the state of the art leaves this gap for this thesis to bridge by considering the users mobility in the deployment of UAV-BS placement based on QL and DQL-based methodologies.

2. The action space of the QL and DQL is discrete and contains a finite number of actions, i.e. one unit towards or against the x, y or z coordinates. So, the UAV-BS agent has to choose its best action from 5 or 7 different actions at each time step. This limitation exists due to the fact that both QL and DQL are value-based RL methods. In value-based RL algorithms, in the exploration phase, the value (quality) of performing each existing action at each state should be evaluated in order to find the best action. Since exploring an unlimited number of actions is not practical, the action space in QL and DQL has to be limited to a finite number of actions, otherwise the agent converges to a local minimum [9].

Considering the above limitations in the literature, this thesis aims for improving the 3D placement of UAV-BSs by employing three RL-based algorithms that add the below extensions to the previous works about UAV-BSs' deployment solutions:

1. Taking into consideration the mobility of users while deploying RL-based algorithms for finding the optimal 3D location of UAV-BSs. This extension will prepare the UAV-BSs' deployment algorithms to be used in real scenarios where the users are highly mobile and their distributions dynamically change.

2. Using a continuous actor-critic deep Q-learning (ACDQL) strategy to solve the location optimization problem of UAV-BSs where the agent can choose its next action from a continuous range $(-k, k)$. The value of k is the maximum distance the UAV-BS can move and it depends on the maximum speed of the UAV-BS. This extension will enable the UAV-BSs to change their locations by any required distance without limiting them to a few predefined actions.

1.4 Contributions

After studying the state-of-the-art approaches about the placement optimization of UAV-BSs and realizing their gaps and limitations, in this thesis we propose to use an advanced type of RL, called continuous ACDQL, for autonomously moving the UAV-BS in the air in order to maximize the performance of the network wherein the action space is continuous and the mobility of users is taken into account.

Key contributions of this thesis can be summarized as follows:

- We propose a reward function for the RL algorithm based on which the agent of RL receives some positive or negative rewards after choosing and performing each of the actions during the training. The proposed reward function is a modified version of the existing commonly-used reward function and it aims to keep the UAV-BS inside the boundaries of the area of interest while it also intends to maximize the users' data rate.

- We develop RL-based schemes for the dynamic positioning of UAV-BSs in the case of users mobility. Unlike the previous works in the literature that assume fixed locations for the ground users, in this thesis, the users mobility is taken into consideration since it significantly impacts the performance results of the UAV-BSs placement algorithms.

- We extend the action space of RL algorithm from discrete to continuous. Different from the previous works in the literature where at each time step, the RL agent can choose the best action from a limited set of predefined actions, in this thesis, we propose to extend the action space to continuous. Therefore, at each time step, the UAV-BS can move by any distance which is only limited by the maximum speed of the UAV-BS.

.

1.5 Research Methodology

In this thesis, we first review the literature to study the solutions already exist for optimizing the location of a UAV-BS and we detect the gaps and missing parts of the existing works. After reviewing the literature and finding the gaps, we propose an air-to-ground channel model so that we can design a wireless communication system and calculate the path loss that ground users experience while receiving data from a UAV-BS. The channel modeling and path loss formulation will help us understand how the propagation environment weakens the wireless signals that reach the users from air. Next, we formulate our optimization problem which aims to move the UAV-BS over a long time period T in a way that the sum data rate of users over T would be maximized. Afterward, since the designed optimization problem is NP-hard, in order to solve it, we propose three methodologies, namely QL, DQL and ACDQL to find the near optimal location of the UAV-BS over the time steps of T that will maximize the users' sum data rate. After proposing our three methodologies, we simulate a cellular network including the users and the UAV-BS. In the simulation, we equip the UAV-BS with each of the proposed strategies and initially place the UAV-BS in one of the corners of the area in the air. Using the proposed strategies, the UAV-BS starts to learn its best movements with the aim of maximizing the sum data rate of users.

1.6 Structure of the thesis

After giving an introduction to this thesis in Chapter 1, we review the state-of-the-art about the placement optimization of UAV-BSs in Chapter 2. Section 2.1 explains the state-of-the-art studies that have used legacy solution for the deployment optimization of UAV-BSs, and Section 2.2 presents the existing AI-based methodologies in the literature for the location optimization of UAV-BSs.

In Chapter 3, we propose a QL-based strategy for the placement optimization of UAV-BSs in the air wherein the mobility information of users is not included as part of the training. Section 3.2.1 presents the air-to-ground channel model that is used in this thesis while in Section 3.2.2, we propose the optimization problem that this thesis aims to solve. Next, we explain the basics of RL and QL and the available formulations in Section 3.3 after which the proposed QL-based strategy is presented in Section 3.4. The simulation results

of the proposed QL-based methodology are illustrated in Section 3.5 which is followed by a conclusion to the chapter in Section 3.6.

Afterwards, in Chapter 4, we propose our DQL-based strategy for the placement optimization of UAV-BSs where users' mobility information becomes part of training. We first give an introduction to this chapter in Section 4.1, followed by explaining the preliminaries of DQL in Section 4.2. Section 4.3 presents our DQL-based method for the deployment of the UAV-BS. The simulation results of the proposed DQL-based strategy is given in Section 4.4. We conclude this chapter in Section 4.5.

Our last strategy for the location optimization of UAV-BSs is proposed in Chapter 5 where mobility information of users is considered in training and the UAV-BS can choose from continuous action space. We first introduce this chapter in Section 5.1 which is followed by reviewing the basics of ACDQL in Section 5.2. Next, in Section 5.3, we propose our ACDQL-based methodology for optimizing the deployment of the UAV-BSs. We evaluate the performance results of the proposed ACDQL-based method in Section 5.4 after which the chapter in concluded in Section 5.5.

Finally, Chapter 6 summarizes the methodologies proposed in this thesis and discusses the research directions and future studies that need to be addressed in a UAV-assisted wireless network.

Chapter 2

Related works

In order to find the optimal 3D location of UAV-BSs under the constraints of network and user benefits, an optimization model needs to be formulated based on the network requirements. As mentioned in the introduction chapter, the 3D location optimization problem of UAV-BSs' is an NP-hard problem with no optimal solution in polynomial time. To tackle the NP-hardness and non-deterministic nature of this optimization problem, there exist several methods in the literature. In one hand, conventional methods such as heuristic ones have been utilized to approximate the best location of UAV-BSs. On the other hand, due to the extensively increasing computational capabilities of processing hardware such as CPUs and GPUs, the AI algorithms have recently been applied in aerial wireless networks in order to solve the UAV-related optimization problems [5, 78]. AI methods that aim to address optimal positioning of UAV-BSs can be classified under three main categories, i.e., supervised learning, unsupervised learning and reinforcement learning. Table 2.1 illustrates a list of key state-of-the-art studies on optimal 3D positioning of UAV-BSs. Each study employs one of the above-mentioned methods. We review both conventional and AI-based solutions in this chapter.

Table 2.1: An overview of the studies about the 3D placement of UAV-BS

Reference	Author(s)	Method	Model	Drones (#)	Drones Altitude
[38]	Kalantari *et al.*	Conventional	Heuristic	Multiple	Dynamic
[82]	Wang *et al.*	Conventional	Heuristic	Multiple	Fixed
[71]	Savkin *et al.*	Conventional	Heuristic	Multiple	Fixed
[14]	Bor-Yaliniz *et al.*	Conventional	Heuristic	Single	Dynamic
[85]	Wang *et al.*	Conventional	Heuristic	Single	Fixed
[26]	Ghanavi *et al.*	AI-based	Q-Learning	single	Dynamic
[50]	Liu *et al.*	AI-based	Q-Learning	Multiple	Dynamic
[40]	Klain *et al.*	AI-based	Q-Learning	Multiple	Dynamic
[90]	Wu *et al.*	AI -based	Deep Q-Learning	Multiple	Dynamic
[29]	Guo *et al.*	AI-based	Deep Q-Learning	Multiple	Dynamic
[103]	Zhong *et al.*	AI-based	Deep Q-Learning	Multiple	Fixed
[7]	Almeida *et al.*	AI-based	Convolutional Neural Network	Multiple	Dynamic

2.1 Methodologies adopting legacy solutions for the optimal placement of UAV-BSs

Heuristic methods have been exploited by various studies to solve the location optimization problem of UAV-BSs. In this section, we review several of these studies that seem promising for the deployment of aerial BSs.

Kalantari *et al.* propose a particle swarm optimization (PSO) meta-heuristic to find the minimum number of UAV-BSs and their 3D placement in a given area with different user densities [38]. Below, we introduce the PSO algorithm to elaborate on the use of PSO for the 3D placement of UAV-BSs.

PSO is a stochastic optimization technique that simulates social behavior of animal swarms such as birds and fish [81]. These swarms conform a cooperative way to find food (solution), and each member in the swarm keeps changing the search pattern according to its own learning experience alongside the learning experience of the other members. This

algorithm starts with a random set of candidate solutions and iteratively tries to improve the candidate solutions with regards to a given measure of quality. The best experience of each candidate as well as the best global experience of all candidates are recorded and the next movement of the candidates is influenced by these items. On the basis of these principles, Kalantari *et al.* [38] formulate a utility function as follows,

$$\underset{\mathbf{w}_i \in \mathbb{R}^{3\times 1}}{\operatorname{argmin}} \sum_{k=1}^{Nsubarea} \sum_{j=1}^{N_{BS}} |N_{U_{BS}} \rho_{j,k} - D_k S_k|, \tag{2.1}$$

where $N_{U_{BS}}$ denotes the maximum number of users that a UAV-BS can serve, N_{BS} is the number of UAV-BS (randomly selected and will improved later), and $N_{subarea}$ is the number of ground subareas with different user densities to be served. In the equation, D_k and S_k are the user density and the area of k-th subarea, respectively. Furthermore, $\rho_{j,k}$ is defined as the ratio of the intersection area between j-th UAV-BS and k-th subarea to the total covered area of j-th UAV-BS. Intuitively, $\rho_{j,k}$ represents the percentage of the mutually covered area by the j-th UAV-BS and k-th subarea. The 3D coordinate of the i-th UAV-BS ($\mathbf{w}_i \in \mathbb{R}^{3\times 1}$) should keep changing for all UAV-BSs in order to minimize the above optimization problem. It is worth to note that we hereby keep using the notation of $\mathbf{w}_i \in \mathbb{R}^{3\times 1}$ or $\mathbf{w}_i \in \mathbb{R}^{2\times 1}$ in all of the optimization problems reviewed throughout this survey as the 3D and 2D (when the UAV-BSs altitude is fixed) coordinates of UAV-BSs, respectively.

The utility function shows the total number of uncovered users which should be minimized. To apply PSO algorithm, a predefined number of initial solutions are generated, i.e. 3D locations of UAVs, and then these solutions are improved in such a way that the utility function is minimized. Across all iterations, the solution which provides the minimum utility is saved as the best global solution. Besides, the lowest utility of the other solutions is stored as the best local. The solutions, i.e., the 3D locations, are updated iteratively based on best global and best local until no further improvement is seen [38]. As another aim is to minimize the number of UAV-BSs, after finding the best locations via PSO, the number of UAV-BSs is minimized by removing the UAVs whose elimination do not affect the quality of the network nor the utility value.

In another study [82], Wang *et al.* investigate how to optimally deploy multiple UAV-BSs with the aim of minimizing the number of UAVs while maximizing their load balance. In fact, considering the battery limitation of UAV-BSs, the variance of the number of served UEs among UAV-BSs should be minimized in order to improve the fairness among UAVs. Key assumption is that all UAVs fly at the same and fixed altitude. Thus, the

optimization problem is formulated on the 2D coordinates of UAVs with the objective of jointly optimizing the number of UAVs and their load balance, and the objective function is formulated as

$$\underset{\mathbf{w}_i \in \mathbb{R}^{2\times 1}}{\operatorname{argmin}} \; H + \frac{1}{H} \sum_{j=1}^{H} \left(\sum_{k=1}^{M} b_{j,k}^{h} - \frac{\sum_{j=1}^{H} \sum_{k=1}^{M} b_{j,k}^{h}}{H} \right)^2, \tag{2.2}$$

where H is the number of UAV-BSs, M is the number of UEs and $b_{j,k}^{h}$ is a binary variable that indicates whether or not the k-th UE is associated with j-th BS. This optimization problem is decomposed into two sub-problems. Initially, a centralized greedy search algorithm obtains the minimum number of UAVs and their sub-optimal positions. This is followed by a distributed motion algorithm to make each UAV autonomously find its optimal position in a continuous space. A closer glance at these algorithms is presented in Fig. 2.1.

The greedy algorithm in [82] consists of five steps. 1) Sufficient number of candidate UAV-BSs are initialized to cover all UEs in a constant predefined altitude. 2) A connection graph is built between UAVs and UEs by calculating the signal-to-interference-plus-noise ratio (SINR) between them. 3) Redundant connections such as one UE being associated with more than one UAV, are deleted. 4) Idle UAVs with no connections are removed. 5) The algorithm searches whether any other UAV-BS can be deleted if it re-associates all its attached UEs to the neighboring UAV-BSs without violating the constraints. At the end of Step-5, an approximation of the minimum number of UAVs are expected to cover all UEs alongside their initial locations.

In the distributed motion algorithm to make UAVs autonomously find their best locations, each UAV models the demand for the optimal coverage as a virtual force field. It is also assumed that all the UAVs can sense their surroundings through target recognition sensors. The virtual force filed for each UAV-BS is a linear function of attractive and repulsive force vectors, which is expressed as

$$\vec{F} = \vec{F}_a + \vec{f}_a + \vec{F}_r + \vec{F}_{obs}, \tag{2.3}$$

where $\vec{F}_a$ represents the attractive forces for connectivity, and it is defined to attract all the UAVs towards the hotspots in the area of interest. This factor is calculated for each UAV-BS based on the distance between the UAV and each hotspot. $\vec{f}_a$ stands for the attractive forces toward UEs instead of $\vec{F}_a$, which is toward the hotspots and its aim is to locate UAVs more precisely in order to cover individual UEs when the distribution of

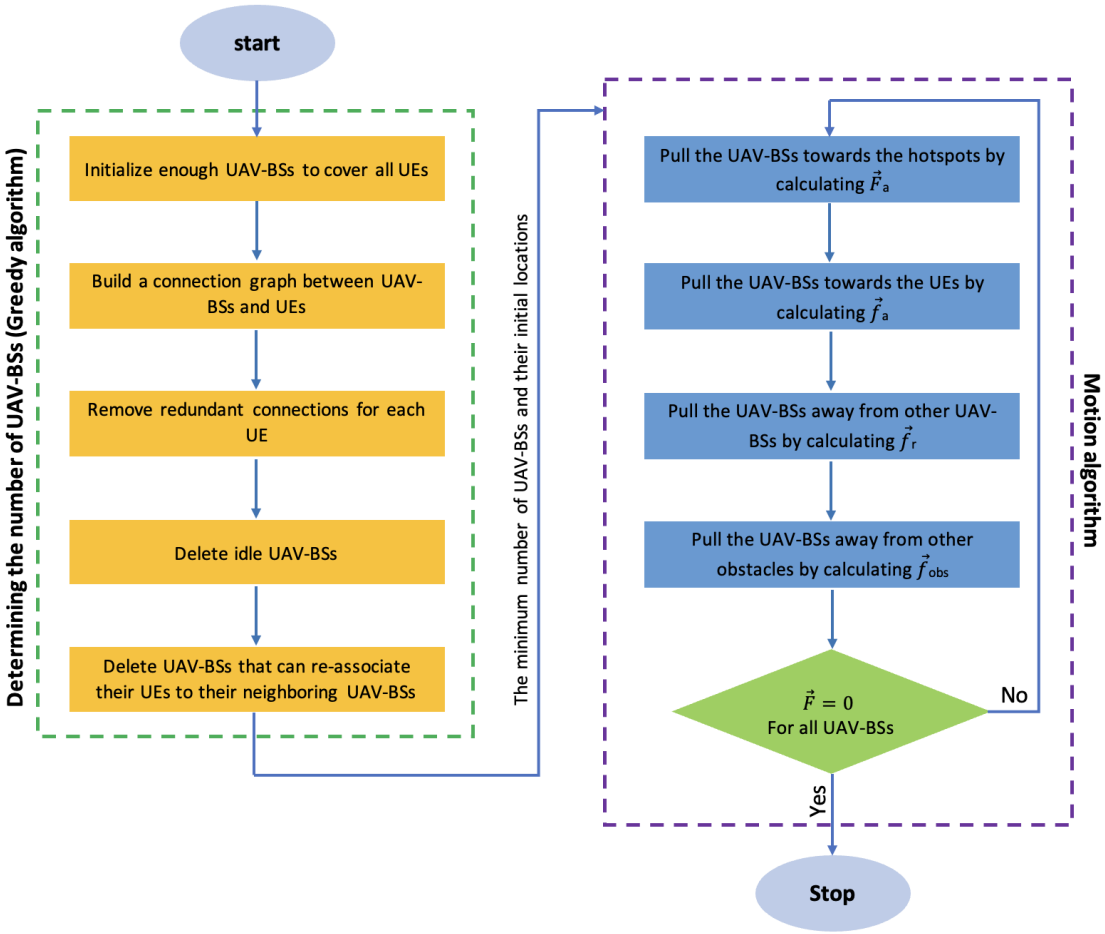


Figure 2.1: The greedy and motion algorithms proposed in [82]

UEs is uneven around the hotspots. This factor is calculated for each UAV-BS based on the distance between each UAV-UE pair. $\vec{F}_r$ denotes the repulsive forces to keep UAVs at desirable distances. While the attractive forces push all the UAVs toward the hotspots, the repulsive forces aim to keep the UAVs away from colliding with each other. The vector $\vec{F}_r$ takes effect when the distance between two UAVs is less than a predefined value. $\vec{F}_{obs}$ denotes the repulsive forces to avoid obstacles and its aim is to avoid UAVs hitting the obstacles. $\vec{F}_{obs}$ will take effect when the distance between a UAV and an obstacle is less than a predefined value, which can be detected by using sensors such as radars, or the geographical information. This factor is calculated based on the distance between each UAV and each obstacle. Considering the above factors, the overall resultant force,

$\vec{F}$, contributes to the optimal placement of UAVs with collision and obstacle avoidance. Every UAV-BS follows the resultant force in (2.3) to move iteratively until the resultant force on every node is zero. The steps of the proposed solution by Wang *et al.* is shown in Fig. 2.1.

Savkin *et al.* [71] propose a greedy algorithm based on Voronoi cells, for finding the optimal location of UAV-BSs during some occasional events when existing traditional ground BS may not be able to handle the excessive demand in the event. The number of UAV-BSs is given in advance, and the aim is to minimize the average UAV-user distance in order to reduce path loss while keeping the UAV-BSs connected to the ground BS. Savkin *et al.* deploy all the UAV-BSs at the same altitude in one horizontal plane. By considering a constant altitude, UAV-BSs' coordinates are considered as the coordinates of a surface. Savkin *et al.* also use a precomputed user probability density function that describes the probability density of users at a particular position at a given time during the occasional event. Such density function is built based on the statistical data obtained from the experience of the previous years.

In order to find the optimal location of UAV-BSs, Savkin *et al.* formulate the optimization problem that minimizes the average UAV-user distance as follows,

$$\underset{\mathbf{w}_i \in \mathbb{R}^{2\times 1}}{\operatorname{argmin}} \int\int_u d^2_{min}(x,y) f(x,y) dxdy, \tag{2.4}$$

where w_i represents i-th coordinates, (x, y) is a particular point in users region, $d_{min}(x, y)$ is the Euclidean distance from point (x, y) to the nearest UAV-BS, and $f(x, y)$ is the weighted user probability density formulated as

$$f(x,y) = r(x,y)\rho(x,y), \tag{2.5}$$

where $\rho(x, y)$ is the user probability density function, and $r(x, y)$ is the data rate requirement of the users at point (x, y).

In order to solve the above NP-hard optimization problem, Savkin *et al.* assume that UAV-BSs can only be deployed at some finite points of the region. In the proposed recursive solution, the UAV-BSs are initially placed at some initial points, based on which the Voronoi diagram is constructed.

Voronoi diagram of a set of n seeds in a space M, partitions M into n regions (cells), one region per seed, such that for each seed, the corresponding region contains all the

points of M that are closer to that seed, than to any other seed [41]. Figure 2.2 is an illustration of Voronoi diagram with the partitioned cells.

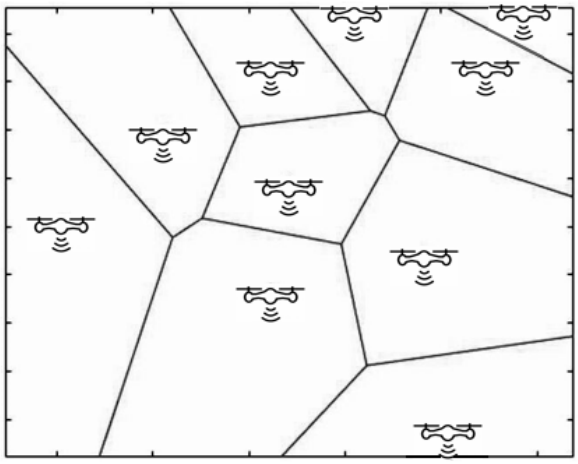


Figure 2.2: Voronoi cells with the UAV-BS over their coverage areas

Taking into account the Voronoi diagram model, in the proposed algorithm in [71], after initialization of the location of UAV-BSs, i.e., the seeds, the Voronoi cells are constructed and the area of interest is partitioned into different Voronoi cells. Afterwards, the centers of Voronoi cells are calculated and considered as the new best locations of UAV-BSs. This algorithm will run recursively until the change in the quality of coverage in two consecutive steps is lower than a predefined threshold. The last centers calculated by the algorithm are considered to be the best location of UAV-BSs that have the lowest packet loss and the best coverage.

In another study in [14], Bor-Yaliniz *et al.* formulated a single UAV-BS placement problem as a 3D placement optimization problem with the objective of maximizing the revenue of the network which is proportional to the number of users covered by the UAV-BSs as a mixed integer non-linear programming (MINLP) problem. The probability of having LoS for a user depends on the altitude of the Drone-BS and the horizontal distance between the user and the Drone-BS. Moreover, for a specific user, the QoS is measured based on the received path loss which is calculated based on the probability of LoS and the 3D Euclidean distance between the user and UAV-BS. The user is served if the QoS is above a predefined threshold.

MINLP problems are optimization problems that have both integer and non-integer

Table 2.2: Optimization objectives used by conventional methodologies for the 3D placement of UAV-BS

Reference	Author(s)	Optimization Objectives
[38]	Kalantari *et al.*	Minimizing the total number of uncovered users
[82]	Wang *et al.*	Minimizing the number of UAV-BSs and maximizing their load balance
[71]	Savkin *et al.*	Minimizing the average UAV-user distance
[14]	Bor-Yaliniz *et al.*	Maximizing the number of served users
[85]	Wang *et al.*	Maximizing the average throughput and the successful transmission probability

variables. Besides, MINLP problems have nonlinear functions in their objective function and/or in their constraints [55]. MINLP problems form a class of very challenging optimization problems that are not solvable for certain inputs [44]. If MINLP problems are forced to contain only linear functions, or restrict them to have no integer values, they can be transformed into NP-hard problems, and interior point optimizer methods are among the state-of-the-art approximators for MINLP problems [13, 79, 21].

In another work [85], Wang *et al.* propose a traffic-aware adaptive deployment algorithm in order to fulfill the instantaneous traffic demands of users when the UAV-BS is unaware of the precise locations of the UEs and only has information about the number of ground UEs in each sector. In Wang *et al.*'s solution, two sectors are assumed on the cell of interest on the ground and there exist two steps: 1-the UAV-BSs first observes the number of UEs in each sector and moves based on a simple majority-vote rule, that is UAV-BS moves a certain distance towards the sector with the maximum number of UEs. So, at the end of step 1, the UAV-BS is placed inside the selected sector $S1$. 2-The authors then continue optimizing the displacement distance of the UAV-BS by further balancing between the two sectors to maximize the average throughput and the successful transmission probability of the network. Wang *et al.* numerically study the optimal value for the UAV-BS displacement via root-finding algorithms, and they finally achieve the below rules:
For maximizing the network average throughput, if the average traffic load in a cell is very small, that happens when the user density is small, the UAV-BS should move towards the center of the sector chosen in step 1. However, if the user density is high in the cell of interest, it means that there are still many users in the other sectors and the UAV-BS's

displacement should be minimal to maximize the average throughput. So, the UAV-BS stays close to the boundary of $S1$ and $S2$. Otherwise, if the aim is to maximize the network successful transmission probability, the following rules are obtained: If the coverage radius of the UAV-BS is not large enough to cover the whole sector $S1$, the optimal strategy for the UAV-BS is to keep its coverage within $S1$. If the coverage radius of the UAV-BS is larger than $S1$, it means that the UAV-BS is capable of not only covering the users inside $S1$, but it can also cover some users of another cluster. In this case, the UAV-BS moves towards another cluster, but still inside $S1$.

In all the proposed conventional solutions for the 3D deployment of UAV-BSs, although the designed optimization problems are NP-hard and not solvable directly, the way they are designed is of critical importance since they are used for measuring how well a solution performs in comparison to the previous solutions with the aim of gradually improving the solution. Table 2.2 reports the optimization objectives of the conventional-based methodologies reviewed in this section.

2.2 Methodologies adopting AI-based solutions for optimal placement of UAV-BSs

While heuristic solutions search for the best possible solution among a set of available solutions depending on the used algorithms, for most of the real-world applications about UAV-assisted cellular networks, the system needs to continuously interact with the environment as the environmental conditions continuously vary and accordingly the set of possible solutions change. Therefore, the heuristic algorithms need to run recurrently in order to adjust with the varying environmental conditions and the new sets of solutions [40]. Thus, the computational overhead of heuristics algorithms dramatically increase as the environmental conditions vary. However, by employing AI and ML methods in dynamic environments, after the model is trained properly, it could adapt the solution for minor changes in the environment without the need to re-run [26]. Since AI-based methodologies do not need to run repeatedly to adapt to dynamic topology, they lead to lower complexity compared to heuristic models and increase the throughput of the network [83].

We now discuss the state-of-the-art studies about the 3D placement of UAV-BSs that have utilized the following AI methods: RL, specifically QL, and DRL, specifically DQL. These are the most popular methods that are used in the 3D placement optimization of UAV-BSs. This is due to the fact that RL and DRL are suitable for the unknown dynamic environments where there is not enough prior knowledge of the environment beforehand and no labeled data is needed in order to train the models. It is worth to note that there are very few studies that apply supervised and unsupervised learning as a potential solution to the 3D positioning of UAV-BSs. For instance, the study in [7] employs a CNN for optimal 3D positioning of UAV-BSs, as it will be discussed later. This section reviews the state of the art location optimization techniques for UAV-BSs based on the AI algorithms.

Ghanavi *et al.* in [26] consider a network of users and terrestrial BSs where the is aim to maximize the overall QoS of the network by adding a single UAV-BS to the system. To do so, QL is empowered in order to find the optimal 3D placement of the new UAV-BS based on the past experiences. The requirement is that the aggregated QoS in the entire network must be above a predefined threshold, QoS_{th}, otherwise the QoS is not acceptable.The objective function of the optimization model introduced by Ghanavi *et al.* is expressed as

$$\underset{\mathbf{w}_i \in \mathbb{R}^{3\times 1}}{\text{argmax}} \sum_{j=1}^{P} \sum_{i=1}^{K} R_{ij} A_{ij}, \tag{2.6}$$

where w_i is a representation of the 3-D coordinates, K is the number of users and P is

the number of BSs, either terrestrial or aerial. A_{ij} is a binary variable and indicated whether user i is assigned to j-th BS. The objective function, R_{ij} denotes the throughput of user i which is served by j-th BS, and calculated based on SINR. Thus, the objective function aims to obtain the optimal position of UAV-BS such that the aggregated throughput of users is maximized.

When a single UAV-BS is used as the agent in RL, the goal can be achieved by maximizing the cumulative reward that the UAV-BS receives. If the QoS becomes lower than a certain threshold, UAV-BS needs to change its position. In the proposed RL-based solution by Ghanavi *et al.*, the state space includes all positions in which the UAV-BS can be located in three dimensions, and the action space is limited to 6 actions. The system receives deterministic rewards based on the action of the UAV-BS in each state. The reward in the t-th time interval is defined as formulated in (2.7). As seen in the equation if the current movement of UAV-BS (the agent's action) leads to a better QoS compared to the previous QoS, the UAV-BS is deemed to obtain a positive reward. Otherwise, the reward value is negative, which means that the UAV-BS are penalized. According to the QL rules, the UAV-BS attempts to choose the best actions based on the current Q-values, and it gradually updates the Q-table based on the obtained reward. Eventually, at the end of the learning phase, the UAV-BS gradually learns how to maximize the QoS and take optimal decisions when network topology changes, as follows,

$$r_t = QoS_t - QoS_{t-1}. \tag{2.7}$$

While in the previous study [26] a single UAV-BS was considered in the network, Liu *et al.* in [50] study multiple UAV-BSs in their system model and propose an RL-based algorithm to obtain the optimal location of UAV-BSs. Liu *et al.* initially use the GAK-means algorithm and partition the ground area into N clusters where each cluster contains a set of K_n users. GAK-means is a combination of the genetic and K-means algorithms in order to tackle the limitation of K-means clustering algorithm. In fact, K-means is prone to being significantly affected by input noise and it can often be attracted to local optimum. However, GAK-means algorithm combines the capacity of genetic operators – where it examines different solutions of the search space – with the advantages of the K-means algorithm thorough which it can obtain global optimum clustering results [35]. First step is to partition the ground users using GAK-means algorithm to K clusters such that the users in a cluster have minimum sum Euclidean distance. These clusters are used to initially deploy K UAV-BSs in random 3D locations. This approach makes use of mean opinion score (MOS) as the measure of the user quality of experience (QoE) which is calculated based on the amount of delay users experience until they receive their entire requested data. Thus, the higher the MOS, the better the service satisfaction. The aim

of Liu *et al.* is to maximize the MOS of ground users by finding the best positioning of UAV-BSs at each time step. Therefore, the objective of the optimization is formulated as

$$\underset{\mathbf{w}_i \in \mathbb{R}^{3\times 1}}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{k_n=1}^{K_n} MOS_{k,n}(r_{k_n}), \tag{2.8}$$

where N is the number of clusters, K_n is the set of users that belongs to cluster n and r_{k_n} is the achievable data rate of user k_n.

A QL algorithm is developed in order to address the above optimization problem [50]. For state representation of each UAV-BS (the agents), $(x_{UAV}, y_{UAV}, h_{UAV})$ which are the 3D coordinates of the UAV-BS are used. Besides, the action space is limited to 7 actions, corresponding to moving 1 unit toward each of the x, y, z coordinates: $(1, 0, 0)$, $(-1, 0, 0)$, $(0, 1, 0)$, $(0, -1, 0)$, $(0, 0, 1)$, $(0, 0, -1)$, or no movement $(0, 0, 0)$. At each timestep (state), each UAV-BS chooses the best action based on its current Q table after which the state of the agent changes and the agent receives a reward determined by the sum MOS of ground users. The proposed reward function is defined as shown in (2.9), in which each UAV-BS attempts to maximize it by choosing the best possible actions as follows,

$$r_t = \begin{cases} 1 & \text{if } MOS_{new} > MOS_{old} \\ -0.1 & \text{if } MOS_{new} = MOS_{old} \\ -1 & \text{if } MOS_{new} < MOS_{old} \end{cases}. \tag{2.9}$$

With the use of RL, each of the UAV-BSs attempts to maximize their cumulative rewards, which leads to maximizing the MOS that is the purpose of the network.

In another study [40], Klaine *et al.* propose a QL-based location optimization method for UAV-BS during emergency situations when a natural disaster completely destroys the cellular network infrastructure while original backhaul of the previous network is still accessible. The proposed QL-based model aims to find the best positions of multiple UAV-BSs with the aim of maximizing the number of served users. Each of the UAV-BSs are considered as an agent in the QL model with a separate Q-table. The state space, action space and reward function that Klaine *et al.* introduce for their model are defined as below:

State space: Defined as the 3D location of the UAV-BS at each time step, where the state space is divided into a grid such that the total number of states becomes finite.

Action space: Each UAV can take seven possible actions, i.e., 1 unit up, down, left, right, forward, backward, and no movement.

Reward function: Since the goal is to maximize the number of served users, the reward is defined as the total number of served user in the current state.

In the solution by Klaine *et al.* [40], the UAVs are initially positioned at random locations. Afterwards, each UAV-BS starts to explore the environment and updates its Q-table to detect the best actions in each state with the aim of obtaining the maximum reward which leads to maximizing the number of served users. The UAV-BSs continuously change their positions to receive more rewards until one of the following stop criteria is met:

− The UAV-BSs have taken actions for the maximum number of iterations based on a predefined number.

− The value of the reward has not improved for a certain number of iterations based on a predefined number.

− The UAV-BS has used all of its resource blocks.

In another study, Wu *et al.* employ a DQL algorithm to find the optimal placement of UAV-BS while users mobility is also considered [90]. In fact, by using DQL, Wu *et al.* solve the dimensional problem of Q-table which happens in dynamic environments. User distribution is considered as a part of the state so that the model can be well adapted to different user distributions. In order to find the optimal location of UAV-BSs, the optimization goal is set to maximizing the average spectral efficiency of each UAV-BS which is defined as the average spectral efficiency of all of its served users. The spectral efficiency of i-th user, served by j-th UAV-BS, is calculated based on the received SINR of the user. Thus, the optimization objective that aims to move each UAV-BS towards the location that has the largest average spectral efficiency Φ_j, is formulated as

$$\underset{\mathbf{w}_i \in \mathbb{R}^{3\times 1}}{\operatorname{argmax}} \ \frac{1}{n} \sum_{i=1}^{n} \Phi(i,j) A_{i,j}, \tag{2.10}$$

where $A_{i,j}$ indicates whether user i is assigned to j-th UAV-BS, and w_i's are the 3D location of the UAV-BSs which can vary within predefined limits (x_{min}, x_{max}), (y_{min}, y_{max}), (h_{min}, h_{max}). Furthermore, $\Phi(i,j)$ is the spectral efficiency of user i served by the j-th UAV-BS.

The proposed DQN by Wu *et al.* consists of a CNN and a QL algorithm where the state space, action space and reward function are defined as follows,

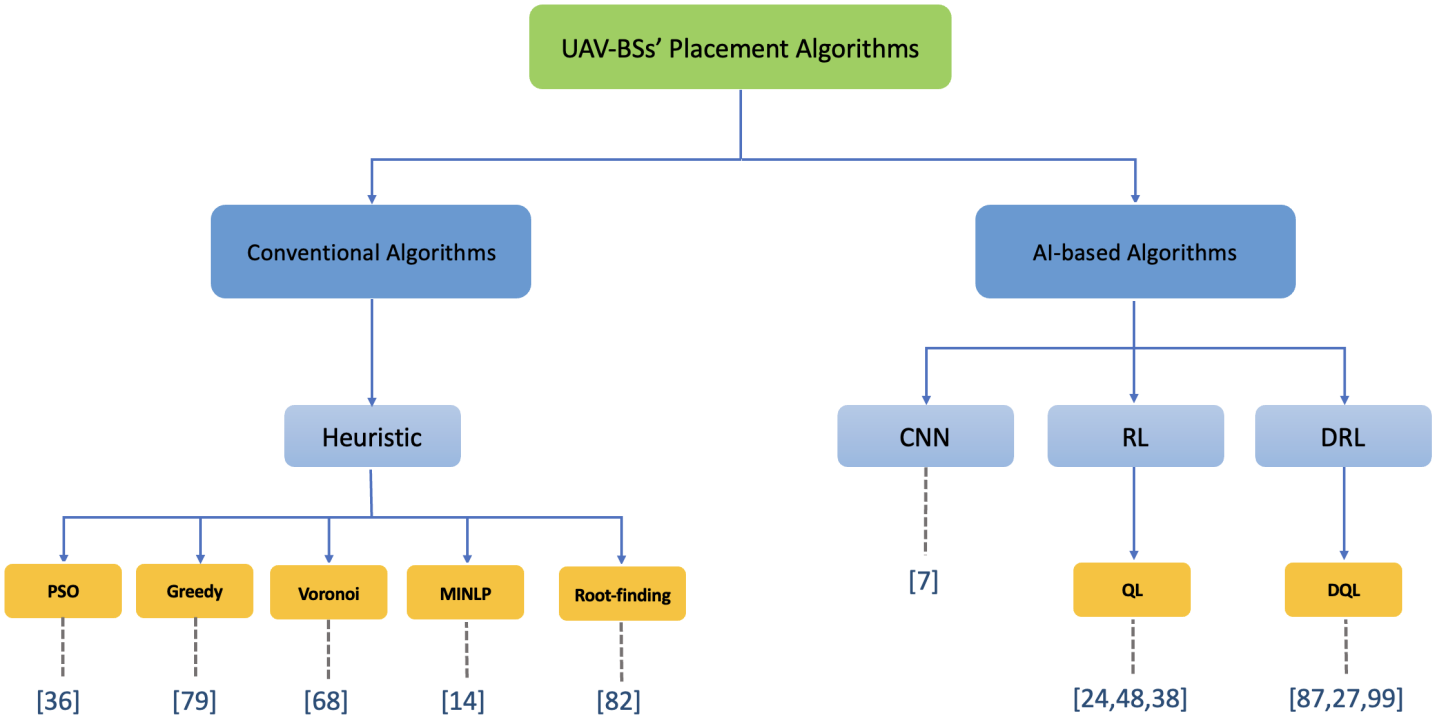


Figure 2.3: Taxonomy of algorithms used in the literature for the location optimization of UAV-BSs

State space: $(x, y, z, u_s, u_{1_{x'}}, u_{1_{y'}}, u_{2_{x'}}, u_{2_{y'}}, \ldots, u_{80_{x'}}, u_{80_{y'}})$, where x, y, z are the 3D coordinates of UAV-BS, u_s is the number of users in the environment, and $u_{i_{x'}}$ and $u_{i_{y'}}$ are the 2D locations of ground users.

Action space: {0, 1, 2, 3, 4, 5, 6}, which corresponds to moving 1 unit in the positive or negative direction of z-axis, x-axis and y-axis or maintaining the current position of the UAV-BS.

Reward function: is defined as the average spectral efficiency of the UAV-BS. So if the action taken by the UAV-BS leads to a high average spectral efficiency, the UAV-BSs receives a high reward which translates to high spectral efficiency.

In the work reviewed above [90], the QoS of users is not taken into account as it is assumed that all users have the same QoS requirements. However, in real-world scenarios, QoS requirements of users may vary depending on various factors. For example, when a disaster happens, rescue team available in the disaster area needs a higher QoS, compared

to other users.Therefore, it is worth to consider various QoS levels while designing an aerial BS-assisted system. With this in mind, Guo *et al.* modify the optimization model in [90] in order to consider multiple QoS levels for users [29]. It is assumed that each user has one of K different QoS requirements, which is measured based on the received SINR. The optimization objective is set to maximizing the spectral efficiency of the whole system, which is the average spectral efficiency of all users, obtained via SINR. Consequently, the goal of the DQN algorithm is to move the UAV-BS towards the location that has the largest average spectral efficiency. Considering that K different QoS levels have been taken into account, the optimization objective in [29], is modified as follows:

$$\underset{\mathbf{w}_i \in \mathbb{R}^{3\times 1}}{\text{argmax}} \ \frac{1}{nm} \sum_{k=1}^{K} \sum_{i=1}^{N_U} \sum_{j=1}^{N_{BS}} \Phi(u_{ik}, j) I_{u_{ik}}, \tag{2.11}$$

where K is the number of different QoS levels, N_U is the total number of users, and N_{BS} is the number of required UAV-BS (calculated based on the total number of users and the maximum capacity of each UAV-BS). $I_{u_{ik}}$ is a binary variable to indicate whether user i requires QoS level k. Additionally, $\Phi(u_{i_k}, j)$ denotes the average spectral efficiency of user i, with QoS level k, served by UAV-BS j. w_i's which are the 3D coordinates of UAV-BSs should be optimized at each time step in order to maximize the overall spectral efficiency in the network.

In order to solve the above NP-hard optimization problem, Guo *et al.* propose a DQL algorithm where the state space is defined as follows [90],

State space: $(x, y, z, u_{ik_x}, u_{ik_y})$, where x, y, z are the 3D coordinates of the UAV-BS, and u_{ik_x} and u_{ik_y} are the 2D coordinates of user i with QoS level k requirement.

Action space and reward function are defined the same as [90].

In another study by Zhong *et al.*, in [103] a DQL algorithm is utilized in order to provide the maximum sum data rate to ground users. The optimization problem aims to achieve the following objective function,

$$\underset{\mathbf{w}_i \in \mathbb{R}^{2\times 1}}{\text{argmax}} \ \sum_{i=1}^{P} C_i(t), \tag{2.12}$$

which aims to maximize the sum data rate of users at each time step t where P is the number of users, and C_i is the data rate of user i, which is calculated based on the received SINR of the user.

In the method reviewed above, a random walk mobility is chosen as the mobility of users where users move in all directions uniformly. When users move at each time step, the UAVs take action to improve the network coverage. Therefore, the objective of the Q network is to observe the locations of both users and UAVs at each time step and derive the action-value function of each UAV-BS. The state space, action space and reward function that Zhong *et al.* introduce in [103] are defined as follows,

State space: (x_{uav}, y_{uav}), which is the 2D location of UAV-BSs, since the altitude of UAVs is assumed to be constant.

Action space: {(1,0), (-1,0), (0,1), (0,-1), (0,0)}, which means UAV-BS can turn right, left, move forward, backward or stay still.

Reward function: The reward function is designed in such a way that it gradually improves the sum data rate of the users: If the action taken by UAV-BS improves the previous sum data rate, the UAV-BS is rewarded by 1 whereas a move towards the direction which decreases the sum data rate penalizes the UAV-BS by -1. Heading towards the locations which have no effect on the sum data rate also penalizes the UAV by -0.2 that helps to prevent UAVs to move ineffectively. Based on these, the reward function is defined as follows

$$r_t = \begin{cases} 1 & \text{if sum rates increase} \\ -0.2 & \text{if sum rates remain the same} \\ -1 & \text{if sum rates decrease} \end{cases}. \tag{2.13}$$

Apart from the studies reviewed earlier, Almeida *et al.* aim to estimate the QoS of aerial wireless networks using a CNN-based QoS estimator [7], which can then be used in heuristic models or other AI-based models to estimate the network QoS, and frequently assess the quality of the network in order to find the optimal placement of UAV-BSs. For example, the QoS estimator can be used for the location optimization of UAV-BSs as a part of the RL algorithm to calculate the reward function. Almeida *et al.* consider the UAV-BS position, user positions and their traffic demand as the input of the proposed CNN model where the CNN learns to estimate the expected QoS corresponding to the input. In order to feed the topology into the CNN model, the ground area is subdivided into smaller fixed-sized zones representing all users on that geographic area. Almeida *et al.* present two matrices to be fed into the CNN model: The former represents the Euclidean distance of the corresponding zone to the closest UAV-BS whereas the latter contains the average

data rate of users that are present in that zone. The corresponding QoS of each input in the training dataset is calculated by the used simulator and considered as the labels. By training the proposed CNN, the parameters of the network, including the filters and weights are gradually learned to be used for predicting the QoS in the future unforeseen network scenarios.

In order to optimize the location of UAV-BSs, all the proposed algorithms first build an optimization problem with a specific objective. As mentioned earlier, these optimization algorithms are all NP-hard with no optimal solution in polynomial time. It was also explained that RL is one of the effective solutions which can obtain a near optimal solution and has lower complexity compared to the conventional methods. The optimization problems proposed in these solutions are of utmost importance because they affect the design of reward functions. Like in heuristic methodologies in which the optimization problems are used for measuring how well a solution is compared to previous solutions, in RL-based solutions, the reward functions are designed according to the optimization problems to help determine how well an action is and to gradually learn the environment.

Table 2.3 reports the optimization objectives of the RL-based methodologies reviewed in

Table 2.3: Optimization objectives and reward functions used by RL-based methodologies for the 3D placement of UAV-BS

Reference	Author(s)	Optimization Objectives	Proposed Reward Function
[26]	Ghanavi *et al.*	Maximizing the users QoS (throughput) which is calculated based on SINR	$QoS_{new} - QoS_{old}$
[50]	Liu *et al.*	Maximizing the users QoE which is calculated based on the received delay	$\begin{cases} 1 & \text{if } MOS_{new} > MOS_{old} \\ -0.1 & \text{if } MOS_{new} = MOS_{old} \\ -1 & \text{if } MOS_{new} < MOS_{old} \end{cases}$
[40]	Klain *et al.*	Maximizing the number of served users	The total number of served user in current state
[90]	Wu *et al.*	Maximizing the average spectral efficiency of each UAV which is calculated based on received SINR of its user (same QoS is considered fro all users)	The average spectral efficiency of the UAV-BS at current state
[29]	Guo *et al.*	Maximizing the average spectral efficiency of each UAV which is calculated based on received SINR of its user (different QoS requirements are considered for users)	The average spectral efficiency of the UAV-BS at current state
[103]	Zhong *et al.*	Maximizing the sum data rate of ground users	$\begin{cases} 1 & \text{if sum rates increase} \\ -0.2 & \text{if sum rates remain the same} \\ -1 & \text{if sum rates decrease} \end{cases}$

this section along with the reward functions proposed in those solutions. This table shows that the reward functions of RL-based solutions are designed according to the optimization objectives of proposed optimization problems that are specific to the UAV-BS use cases and scenarios.

As reviewed in this chapter, there are a number of works that have studied the problem of optimizing the location of UAV-BSs in the air. Amongst all types of solutions we explained, heuristic methods are not suitable for dynamic environments because they have to re-run repeatedly for every minor change in the network. AI-based methodologies on the other hand, are appropriate for dynamic environments since during the training, they learn how to adopt to the minor changes of the environment with no need to re-run. However, the state-of-the-art AI-based solutions neglect the mobility of users while proposing the deployment algorithms although users mobility significantly impacts the performance results of the proposed algorithms. Moreover, as seen throughput this chapter, the action space of RL-based algorithms used by the state-of-the-art is limited to a few actions, i.e. maximum 5 or 7 actions. Therefore, UAV-BSs have to choose their actions from very few number of actions at each time step. Thus, in the next chapters of this thesis, we aim to cover the above two mentioned gaps in the literature by considering the users mobility as well as extending the action space of RL to continuous. Our continuation to the state-of-the-art is developed step-by-step as follows:

- A QL-based positioning strategy is first proposed that uses a modified version of the existing commonly-used reward function. This reward function aims to keep the UAV-BS inside the boundaries of the area of interest while it also intends to maximize the users' data rate. The proposed QL-based model has discrete state and action spaces and due to the limitations of QL, does not include the mobility information of users in the process of learning the environment.

- As an extension to the limitations of the proposed QL-based model, a DQL-based strategy is proposed to involve users' mobility information when we train the UAV-BS agent. The state space expansion from discrete to continuous improves the system performance to a great degree. The action space, however, is still discrete in the proposed DQL-based model because DQL is a value-based RL algorithm wherein having unlimited number of actions is infeasible.

- For further improving the performance, a continuous ACDQL-based algorithm is proposed. In the proposed ACDQL, a combination of two functions, i.e. a value-based function and a policy-based function, is used in order to have a continuous action space from which the UAV-BS can choose its movements at each time step. Our

simulation results show that changing the action space from discrete to continuous improves the network performance remarkably.

Chapter 3

Q-learning-based deployment of UAV-BSs

3.1 Introduction

In this chapter, a QL-based strategy is proposed for solving the location optimization problem of UAV-BSs. As mentioned in Chapter 1, one of the main contributions of this thesis is proposing a new reward function for the RL-based agent of UAV-BSs which is a modified version of a reward function previously used in the literature by several researchers. The goal of our reward function is to maximize the data rate of ground users while keeping the UAV-BS inside the predefined boundaries. We propose this newly-designed reward function in this chapter while it will be exploited in the next chapters as well. After proposing our reward function, we propose a QL-based algorithm that uses the designed reward function for maximizing the data rate of ground users. The proposed QL-based methodology considers the mobility of ground users which is another contribution of this thesis since the majority of previous works assume fixed locations for the users. Our simulation results show that the performance of the proposed model outperforms the random-based movement of the UAV-BS in terms of users data rate, packet loss and transmission delay. However, the results of the QL-based strategy is worse than another baseline in which the the UAV-BS moves in accordance with the mobility pattern of users. In fact, since we define the mobility pattern of ground users for the simulation, we set the exact mobility model for the UAV-BS as well, to see how the performance results improve if the UAV-BS can adjust its positions according to the movement pattern of users.

3.2 System model

Before proposing our RL-based strategies, namely QL in this chapter, and DQL and ACDQL-based methodologies in the next chapters, we should first investigate on how a UAV-enabled wireless system is modeled so that we can use it in our proposed models and later in our simulation. In other words, we should study how the wireless signals weaken in an air-to-ground (A2G) communication system, how to calculate the path loss, how the users' data rate is calculated, etc. Therefore, this section explains the channel model we define for this thesis along with the channel modeling and the problem formulation.

Our system model is illustrated in Fig. 3.1. This thesis considers a scenario where the ground BS is failed and a UAV-BS is supposed to provide downlink Internet connectivity to users on the ground. The area where the users are located is a $D \times D$ km^2 area where the initial locations of users are randomly selected over the entire area. At initial step, the UAV-BS positions randomly in the air at the fixed altitude h. There is n number of users in our model and the 2D coordinates of each ground user i is denoted as (x_i, y_i), while $(x_{\text{uav}}, y_{\text{uav}}, h_{\text{uav}})$ is the 3D coordinates of the UAV-BS. The environment under the study is assumed to be an urban environment where there are both LoS and NLoS connections. The users are mobile and their mobility model follows the Gauss-Markov mobility model. In Gauss-Markov mobility model, an initial random speed and direction is assigned to each user. The speed and direction of users change at specified intervals of time where at time interval t the speed and direction is calculated on the basis of the speed and direction at time interval $t-1$ [15]. Table. 3.1 presents all the notations in our proposed system.

The UAV-BS is considred as an RL agent and equipped with an RL-based algorithm. At each time step t, the UAV-BS selects its best movement on x and y coordinates separately with the aim of maximizing the users data rate.

3.2.1 Channel model

Due to the different characteristics of air section, A2G channels are different from the channels established on the ground. The communication channel between the UAV-BS in altitude h_{uav} and UEs on the ground is modeled with path loss which depends on the probability of LoS and that of NLoS between UEs and the UAV-BS.

Table 3.1: Table of notations used in our proposed system model

Parameter	**Definition**
$D \times D$	dimension of the area (m)
h_{uav}	UAV-BS's altitude (m)
N	number of users
(x_i, y_i)	2D coordinates of i-th user
$(x_{\text{uav}}, y_{\text{uav}}, h_{\text{uav}})$	3D coordinates of UAV-BS
k	maximum speed of UAV-BS (mps)
(a, b)	environment type variables
θ_i	elevation angle between i-th user and UAV-BS
$P_{i(\text{LoS})}$	probability of LoS between i-th user and UAV-BS
$P_{i(\text{NLoS})}$	probability of NLoS between i-th user and UAV-BS
r_i	horizontal distance between i-th user and UAV-BS (m)
f_c	carrier frequency (Hz)
c	speed of light(mps)
$\delta_{\text{LoS}}, \delta_{\text{NLoS}}$	additional environmental path loss for LoS and NLoS (dB)
R_i	downlink data rate of i-th user
p_i^t	transmission power from the UAV-BS to i-th user (dBm)
p_i^r	received power from the UAV-BS to i-th user (dBm)
p_{uav}	total transmission power of UAV-BS (dBm)
B	total bandwidth of UAV-BS (bit)
γ_i	SNR received from UAV-BS to i-th user (dB)
σ	noise power (dBm)
G_i	power gain from UAV-BS to i-th user (dB)
g_i^{tx}	transmitting antenna gain from UAV-BS to i-th user (dB)
g_i^{rx}	receiving antenna gain from UAV-BS to i-th user (dB)
d_0	far field reference distance (m)

Since UAV-BSs are deployed in 3D space above the ground, there is a high probability of establishing LoS connections with ground users. The probability of LoS between the UAV-BS in altitude h_{uav} and i-th ground user can be formulated as [4]

$$P_{i(\text{LoS})} = \frac{1}{1 + a\, exp(-b[\theta_i - a])}, \tag{3.1}$$

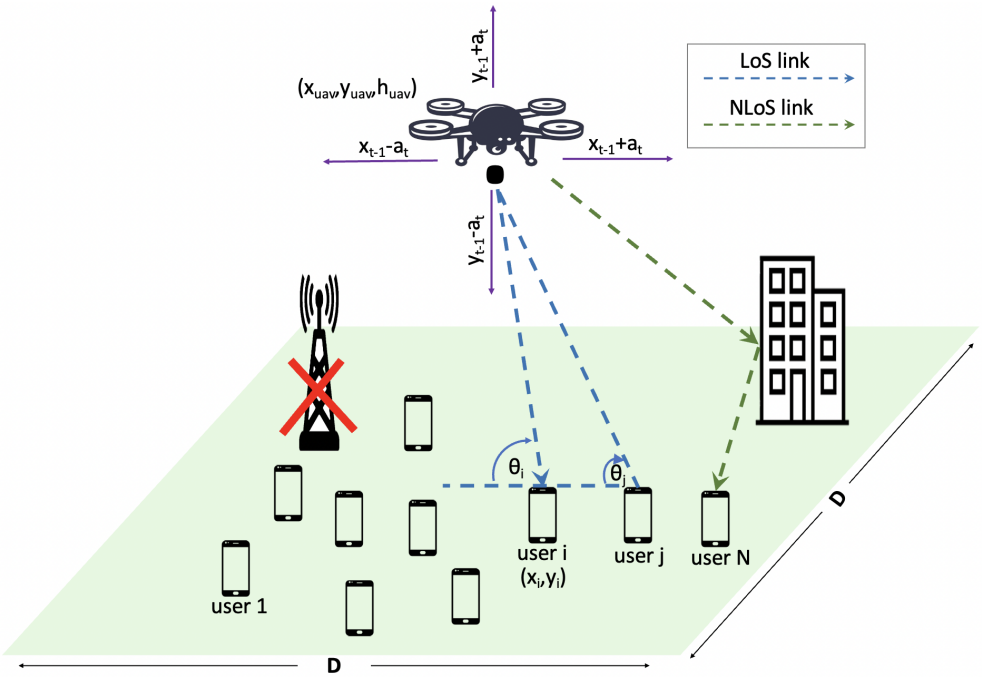


Figure 3.1: System model (all users are mobile)

where a and b are set according to the environment type (rural, urban or dense urban) and they represent the ratio of built-up land area to the total land area, and the mean number of buildings per unit area, respectively. θ_i is the elevation angle between the UAV-BS and i-th user and is calculated by

$$\theta_i = \arctan(h_{\text{uav}}/r_i), \tag{3.2}$$

where r_i is the horizontal distance between the i-th user and the UAV-BS which is obtained as

$$r_i = \sqrt{(x_{\text{uav}} - x_i)^2 + (y_{\text{uav}} - y_i)^2}. \tag{3.3}$$

The path loss associated with LoS connections is calculated as [4]

$$L_{i(\text{LoS})} = 20\, log(\frac{4\pi f_c r_i}{c}) + \delta_{\text{LoS}}, \tag{3.4}$$

where f_c is the carrier frequency, c is the speed of light and δ_{LoS} is the additional environment-dependant path loss for LoS connections.

Due to the existence of obstacles in the environment such as tall buildings, trees, etc, especially in urban and dense urban environments, some signals from the UAV-BS reach the users through NLoS links. The probability of NLoS between the UAV-BS and i-th user is given as [4]

$$P_{i(\text{NLoS})} = 1 - P_{i(\text{LoS})}. \tag{3.5}$$

And the path loss connected with NLoS connections is obtained as [4]

$$L_{i(\text{NLoS})} = 20\, log(\frac{4\pi f_c r_i}{c}) + \delta_{\text{NLoS}}, \tag{3.6}$$

where δ_{NLoS} is the additional environment-dependant path loss associated with NLoS connections.

Finally, the average path loss from the UAV-BS to the i-th user is written as

$$L_{i(\text{avg})} = P_{i(\text{LoS})} \times L_{i(\text{LoS})} \; + \; P_{i(\text{NLoS})} \times L_{i(\text{NLoS})}. \tag{3.7}$$

3.2.2 Optimization problem formulation

As reviewed in Section 3.2.1, the path loss of the connections established between UAV-BS in the air and the ground users depends on the probability of LoS and NLoS which is impacted by the 3D location of UAV-BS. Since the users are mobile and their distribution changes over time, the UAV-BS should continuously change its location and move towards the locations that maximize the system throughput, where the system throughput is defined as the achieved data rate of ground users. Therefore, our objective in this thesis is to move the UAV-BS in a path that maximizes the users' sum data rate over time steps t of time period T, that can be written as

$$\underset{x_{\text{uav}}, y_{\text{uav}}, h_{\text{uav}}}{\text{argmax}} \sum_{t=1}^{T} \sum_{i=1}^{N} R_i(t), \tag{3.8}$$

$$s.t. \sum_{i=1}^{N} p_i^t = p_{\text{uav}}, \tag{3.9}$$

$$\begin{aligned} 0 \leq x_{\text{uav}} \leq x_{\max}, \\ 0 \leq y_{\text{uav}} \leq y_{\max}. \end{aligned} \tag{3.10}$$

where the constraint in Eq. 3.9 guarantees that the total allocated power from the UAV-BS to the users cannot exceed the transmission power of the UAV-BS which is denoted by p_{uav}. Also, the constraints in Eq. 3.10 assure that the UAV-BS cannot fly out of the predefined boundaries.

R_i is the data transmission rate between the UAV-BS and i-th user that can be calculated as [91]

$$R_i = \frac{B}{N} \log_2(1 + \gamma_i), \tag{3.11}$$

where B is the total bandwidth of the UAV-BS which is equally allocated to the users. γ_i is the signal-to-noise ratio (SNR) of the signal received from the UAV-BS to i-th user and can be expressed as [91, 8]

$$\gamma_i = \frac{p_i^r \, G_i}{\sigma^2}, \tag{3.12}$$

where p_i^r is the received power from the UAV-BS to the i-th user which is calculated by $p_i^r = p_i^t - L_{i(\text{avg})}$. In order to have a guaranteed quality of service for the user, it is assumed that $L_{i(\text{avg})}$ does not exceed the transmission power p_i^t. Moreover, σ is the noise power and G_i is the power gain from the UAV-BS to the i-th user that can be calculated as [91]

$$G_i = \frac{g_i^{tx} \, g_i^{rx} \, (\frac{c}{f_c})^2}{16\pi^2 (\frac{d_i}{d_0})^2}, \tag{3.13}$$

where g_i^{tx} and g_i^{rx} are transmitting and receiving antenna gains from the UAV-BS to the i-th user. d_0 is a far field reference distance while d_i is the distance between the i-th user and UAV-BS which is calculated as

$$d_i = \sqrt{h_{\text{uav}}^2 + (x_{\text{uav}} - x_i)^2 + (y_{\text{uav}} - y_i)^2} \tag{3.14}$$

The proposed objective function in 3.8 which aims to maximize the users' sum data rate over long run T is a mixed integer non-linear programming (MINLP) formulation since it is an optimization problem with both integer and continuous variables. In [20, 18], it is proven that MINLP problems are NP-hard with no solution in polynomial time. Therefore, in order to solve this NP-hard problem that maximizes the sum data rate of ground users in a UAV-assisted cellular network, we will present three solutions, i.e. QL, DQL and ACDQL-based models, in the next three chapters.

Now that we have defined our objective function in 3.8 and we have determined our channel model, in this chapter, we present a QL-based model in order to resolve the presented objective function. The proposed objective function and the channel model will be used in the next chapters as well, i.e. Chapter 4 and Chapter 5, where two more advanced solutions, i.e. DQL and ACDQL-based models are proposed in order to optimize the location of UAV-BSs.

3.3 Preliminaries for Q-learning

3.3.1 Reinforcement learning

RL is a field of ML in which an agent attempts to learn how to behave in an environment when its only feedback from the environment is a scholar reward signal [89]. The goal of the agent is to perform a set of actions that lead to receiving maximum reward signals in a long run. The elements of RL can be formalized using Markov decision process (MDP) which consists of states (s), actions (a) , transitions between the states and the reward function. A system is said to be Markovian if the result of an action does not depend on the previous actions and states, but only depends on the current state which is formulated as [89]

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ...) = P(s_{t+1}|s_t, a_t). \tag{3.15}$$

Indeed, according to 3.15, in the random environment of RL, the current state captures the information of the previous states and has enough information to make decision. There are two more definitions which are used in RL, i.e., policy and value. A policy determines which action should be taken in each state and the value of a state, also known as cost of the state, is the cumulative reward that is expected when starting from that state until reaching the target. The goal of RL is to find the best policy which receives the most reward [75]. However, in RL, the impacts of actions are not immediately understood and

they can be much delayed until reaching the target state [89]. According to the Bellman equation, in order to calculate the cost of a state, a discounted value function is defined as [67]

$$V(s_t) = \underset{a_t}{\text{argmax}}(r(s_t, a_t) + \gamma V^*(s'_{t+1})), \tag{3.16}$$

where $r(s, a)$ is the immediate reward the agent receives by performing action a that moves the agent to state s', and $V^*(s')$ is the optimal cost of the state s'. In the same equation, γ is the discount factor to quantify the importance of future rewards and it is a number between zero and one. If the discount factor is zero, the agent only learns to choose the actions with immediate rewards and neglects the future rewards. However, if the discount factor is one, the summation in Eq. 3.16 could be infinite and the algorithm does not convergence. Therefore, discount factor is a parameter that needs to be carefully selected according to the environment. The agent takes random actions and explore random states by trial and error and gradually updates the value of states. However, as MDP is a stochastic process, Bellman equation is adopted as below to include the environment randomness [67]:

$$V(s_t) = \underset{a_t}{\text{argmax}} \left\{ R(s_t, a_t) + \gamma \sum_{s'} P(s_t, a_t, s'_{t+1}) V^*(s'_{t+1}) \right\}, \tag{3.17}$$

where $P(s, a, s')$ is the probability of moving from state s to state s' by performing action a. Figure 3.2 illustrates RL and how the agent and environment interact with each other.

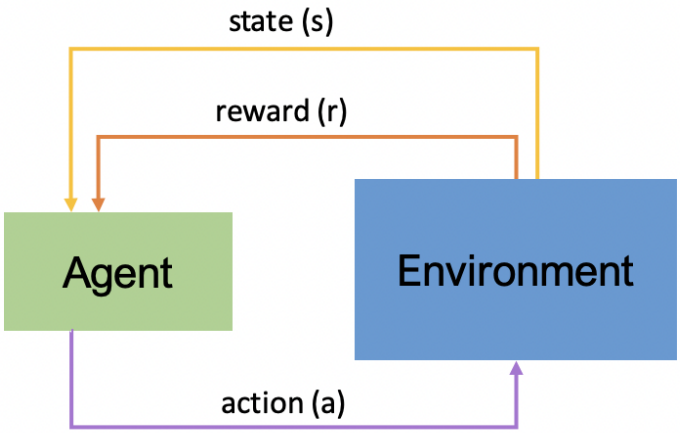


Figure 3.2: Reinforcement learning

3.3.2 Q-learning

QL is a model-free RL which tries to learn based on the consequences of actions without any need to build the domain map beforehand [86]. In QL, rather than analyzing the value of states, the quality of an action in a state is evaluated. Indeed, instead of calculating $v(s)$, $Q(s, a)$ is evaluated, which means the expected discounted reward for executing action a in state s [86]. So, the Bellman equation, used for RL, is adjusted as below to obtain Q function [65]:

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma \underset{a'_{t+1}}{\operatorname{argmax}} Q(s'_{t+1}, a'_{t+1}), \tag{3.18}$$

where $Q(s', a')$ is the Q value of the destination state after performing action a'.

At each timestep, the Q-values are stored and updated in a table called Q-table in which the columns are the actions and the rows are the states. Fig. 3.3 depicts how in QL, the states and actions are mapped to Q-values through a Q table.

Considering that the environment is random, the Q-value associated with state s and action a might be different from the previous Q-values. Therefore, temporal difference (TD) is introduced in QL in order to present the difference between the new predicted Q-value $Q_t(s, a)$ and the old one $Q_{t-1}(s, a)$. Accordingly, TD is formulated as below [52],

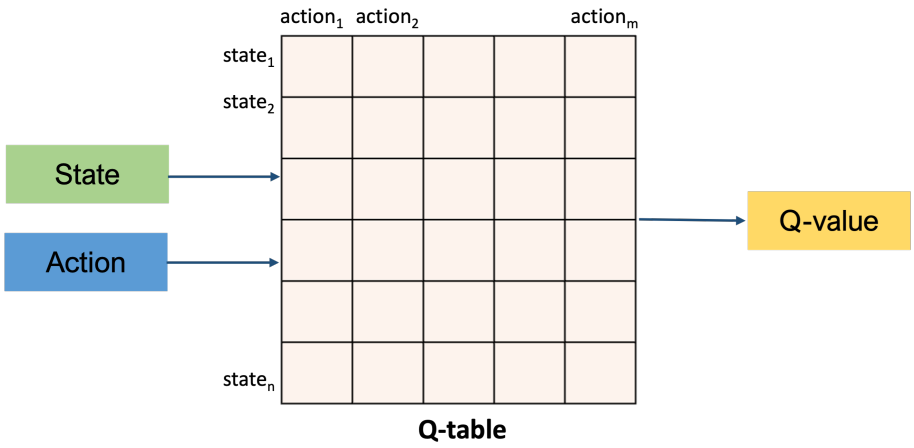


Figure 3.3: Q-learning

$$TD(s,a) = R(s,a) + \gamma \underset{a'}{\operatorname{argmax}} Q(s',a') - Q_{t-1}(s,a). \tag{3.19}$$

Taking into account the impact of TD in Q-values, the Q function is modified as below:

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha TD_t(a,s), \tag{3.20}$$

where α is the learning rate. Considering the two formulations above (i.e., TD and Q functions), the final Q function is derived as

$$\begin{aligned} Q_t(s,a) &= Q_{t-1}(s,a) + \alpha(R(s,a) \\ &+ \gamma \underset{a'}{\operatorname{argmax}} Q(s',a') - Q_{t-1}(s,a)). \end{aligned} \tag{3.21}$$

The agent at each step attempts to choose the action with maximum expected Q-values. After choosing an action, the state of environment changes and the agent receives a reward that updates the Q-table. This process continues and will cause the agent to gradually learn the environment and reach its target.

Moreover, the agent can stick with the experience it has gained so far in order to choose the best actions at each step, which is known as the **exploitation** strategy [99]. However, exploitation strategy seems not always viable especially when the agent has not explored the environment sufficiently and does not have enough experience. So, the agent

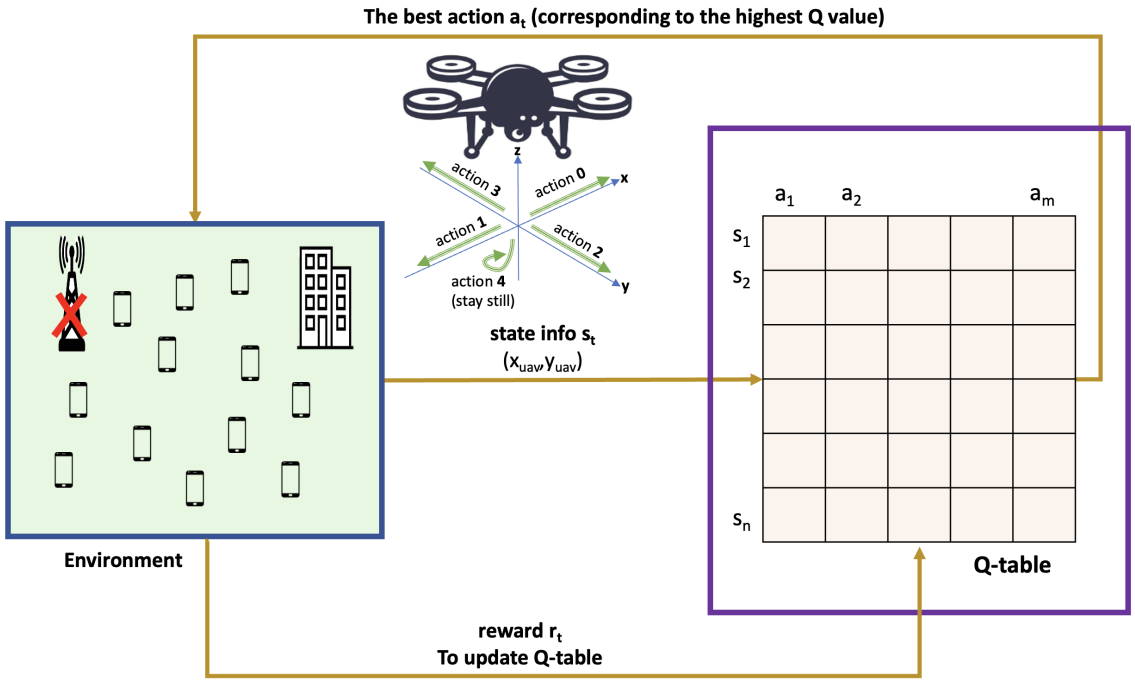


Figure 3.4: UAV-BS deployment with QL

can follow the **exploration** strategy where it tries to experience something new without being committed to using old experience with the assumption of other promising solutions will be found [25]. One of the renown strategies that combine exploitation and exploration in Q-learning is the **ϵ-greedy** strategy in which the agent occasionally pursues random exploration with the probability of ϵ while it exploits the optimal action most of the time with the probability of $1 - \epsilon$ [77].

3.4 The proposed QL-based deployment of UAV-BSs

In this section, we propose a QL-based strategy that takes into account the real-time location of the UAV-BS in the sky, in order to find the next optimal location of the UAV-BS with the aim of maximizing the sum data rate of users. In our proposed QL-based algorithm, the UAV-BS is considered as the agent of QL whose aim is to maximize the Q-values associated with each pair of state and action $Q(s, a)$. The agent follows exploration and exploitation strategies to learn the environment. At each time step t, following upon a transition $(s_t, a_t, r_t, s'_{t+1})$, the Q-table is updated according to the equation 3.21 until the

Q-values converge. Fig. 3.4 illustrates our QL-based strategy with an emphasis on the interaction between the UAV-BS agent and the environment so as to update the Q-table and choose the best actions. The system model in the proposed QL-based strategy follows what proposed in Section 3.2 and Fig. 3.1.

As we presented in our system model, in Section 3.2.2, on the one hand, the system objective is to maximize the sum data rate of all users. On the other hand, QL aims to maximize the cumulative reward that the agent receives after choosing the actions over a long run. That being said, the reward function, used by our proposed QL-based

Algorithm 1 QL-based optimal placement of UAV-BSs

for each episode **do**
  initialize users randomly
  initialize UAV-BS in one of the 3D corners with altitude h
  for each time t step in episode **do**
    observe $s_t = (x_{\text{uav}}, y_{\text{uav}})$
    $a_t = \begin{cases} \text{random number} & \text{with the probability of } \epsilon \\ argmax(Q[s_t]) & \text{with the probability of } 1-\epsilon \end{cases}$
    Move UAV-BS according to the chosen action a_t
    observe new state $s'_{t+1} = (x'_{\text{uav}}, y'_{\text{uav}})$
    for every user n **do**
      calculate R_n
    end for
    $r_t = \begin{cases} -\mathcal{R} & \text{if} \quad x_b(t) \notin [0, x_{max}] \quad \text{or} \quad y_b(t) \notin [0, y_{max}] \\ 1 & \text{if} \quad \sum_{n=1}^{N} R_n(t) \text{ increases} \\ -1 & \text{if} \quad \sum_{n=1}^{N} R_n(t) \text{ decreases} \\ -0.2 & \text{if} \quad \sum_{n=1}^{N} R_n(t) \text{ does not change} \end{cases}$
    $Q_t(s,a) = Q_{t-1}(s,a) + \alpha(r_t + \gamma \max Q_{t+1}(s') - Q_{t-1}(s,a))$

    if $\epsilon \geq$ epsilon_min **then** $\epsilon = \epsilon \times$ epsilon_decay

    else $\epsilon =$ epsilon_min
    end if
  end for
end for

methodology, must be designed to maximize the sum data rate of all users over a long run.

In the proposed QL-based strategy, the agent, state space, action space and reward function are described as follows:

Agent: The UAV-BS in the air is considered as the QL agent.

State Space: In QL, the number of states should be limited in order to keep the Q-table effective and practical [36]. Since the proposed environment is large with many users, the state space can only include the position of UAV-BS which is expressed as follows:

$\mathbf{s}_t = (x_{\text{uav}}(t), y_{\text{uav}}(t))$

where x_{uav} and y_{uav} can only take discrete values $\{1, 2, ..., max_{\text{x\_index}}\}$ and $\{1, 2, ..., max_{\text{y\_index}}\}$ accordingly.

Action Space: The proposed action space, from which the UAV-BS chooses action at each time step, consists of five different action codes, i.e. 0, 1, 2, 3, 4. These action codes correspond to moving towards the positive direction of x axis, the negative direction of x axis, the positive direction of y axis, the negative direction of y axis and stay in the same coordinates with no movement, accordingly. The distance that UAV-BS moves at each time step towards or against the x and y axis depends on the speed of UAV-BS. Thus, a UAV-BS at a speed of $g\ m/s$, moves g meters at each time interval if the chosen action is not 4.

$\mathbf{a}_t = \{0, 1, 2, 3, 4\}$

Reward Function: As the objective is to maximize the sum data rates of users, the reward function in our model is formulated as follows where $x_{\text{uav}}(t)$ and $y_{\text{uav}}(t)$ are the x and y coordinates of the UAV-BS at time step t.

$$r_t = \begin{cases} -\mathcal{R} & \text{if} \quad x_{\text{uav}}(t) \notin [0, x_{\max}] \text{ or } \ y_{\text{uav}}(t) \notin [0, y_{\max}] \\ 1 & \text{if} \quad \sum_{i=1}^{N} R_i(t) \text{ increases} \\ -1 & \text{if} \quad \sum_{i=1}^{N} R_i(t) \text{ decreases} \\ -0.2 & \text{if} \quad \sum_{i=1}^{N} R_i(t) \text{ does not change} \end{cases}, \tag{3.22}$$

According to our proposed reward function –which is a modified version of that in [103]– if the UAV-BS moves outside of the area of interest, a significantly low reward (-$\mathfrak{R}$) is given to the agent. Otherwise, if the action taken by the UAV-BS at time t improves the total data rate of users, the agent receives a positive reward of +1. In contrast, if the action makes the total users' data rate drop, the agent gets a negative reward of -1. Moreover, if the current action makes no difference in the data rate, a negative reward of -0.2 is given to the agent in order to prevent the UAV-BS from doing unnecessary movements that wastes its limited energy.

The steps of the proposed QL for 3D placement of an UAV-BS are illustrated in Algorithm 1.

3.5 Simulation Results

In our simulation, to comply with the related work, we consider a square area of 1×1 km^2 [50][51] where 50 users are randomly distributed on the ground. As the majority of works considered a fixed altitude for the UAV-BS, in this thesis also, the UAV-BS flies at the fixed altitude of $100m$ [49, 94] and is initially positioned in random horizontal coordinates above the ground. However, the simulation can be easily performed with a flexible altitude where we only need to add the altitude of the UAV-BS to our state space. Nevertheless, since the aim of this thesis is to evaluate the impact of users' mobility and continuous action space, the altitude of the UAV-BS is assumed fixed for all the three proposed methodologies and we still can evaluate our contributions and compare the performance results of the proposed models. The users move according to the Gauss-Markov mobility model where their speed varies between 2 m/s and 8 m/s. The maximum speed of the UAV-BS is set to 6 m/s that determines the maximum distance the UAV-BS can moves towards or against the x and y coordinates at each time step. Other parameters related to our simulation environment are listed in Table 3.2 which are selected according to [90, 91]. We run our simulation for 600 episodes where the initial positions of users and the UAV-BS change at the beginning of each episode. Each episode of the simulation consists of 200 time steps ($T = 200s$).

In order to compare the results of our proposed QL-based methodology with baselines, we consider below two conventional algorithms for moving the UAV-BS:

1. **Random Action:** In this algorithm, the UAV-BS's action is randomly chosen in each time step. Thus, the UAV-BS moves in random directions to cover users on

Table 3.2: Environmental parameters of the simulation (based on [90, 91])

Parameter	**Value**
urban environmental variable a	11.95
urban environmental variable b	0.136
carrier frequency f_c	2GHz
additional LoS environmental path loss δ_{LoS}	2.09
additional NLoS environmental path loss δ_{NLoS}	3.75
UAV-BS maximum transmit power p_{uav}	24dBm
UAV-BS Bandwidth B	20MHz
transmitting antenna gain g_i^{tx}	1
receiving antenna gain g_i^{rx}	1
noise power σ^2	-90dBm
far field reference distance d_0	1
altitude of UAV-BS h_{uav}	100m

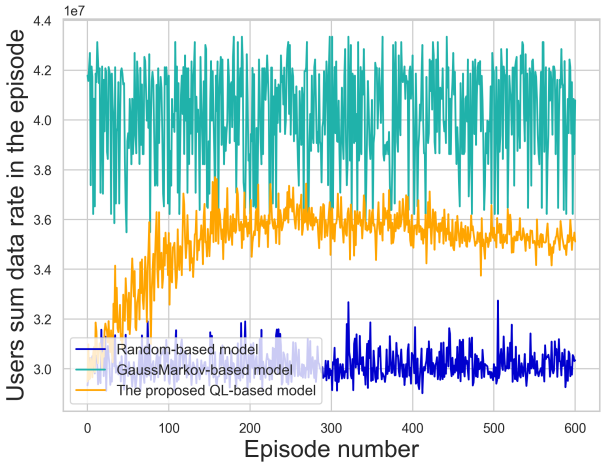
different locations on the ground.

2. **Gauss Markov mobility model:** In this model, the same mobility model we use for UEs is adopted as the mobility model of UAV-BS. Therefore, in this baseline, the UEs and UAV-BS follow identical mobility models with the same parameters, e.g. direction and speed.

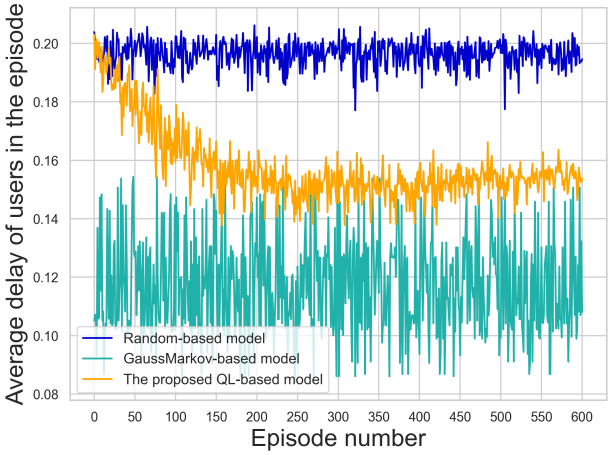
To test the optimality of the solution, we first compare the achieved sum data rate of users under the proposed QL-based strategy with the sum data rate obtained from the two baselines. In order to do so, the sum data rate of users is recorded in each episode which is defined as the summation of the data rates of all users in different time steps of that episode.

Fig. 3.5a compares the cumulative data rate of users under the different methodologies. As opposed to the random movement of the UAV-BS, the proposed QL-based method can successfully improve the data rate of users by selecting the best action to be taken by the UAV-BS at each time step until it converges after around 250 episodes. After 250 episodes, this proposed model cannot further improves the data rate. However, by looking at the data rate results obtained from Gauss-Markov-based model, we realize that the QL model has not reached the maximum achievable data rate. In other words, if the UAV-BS moves

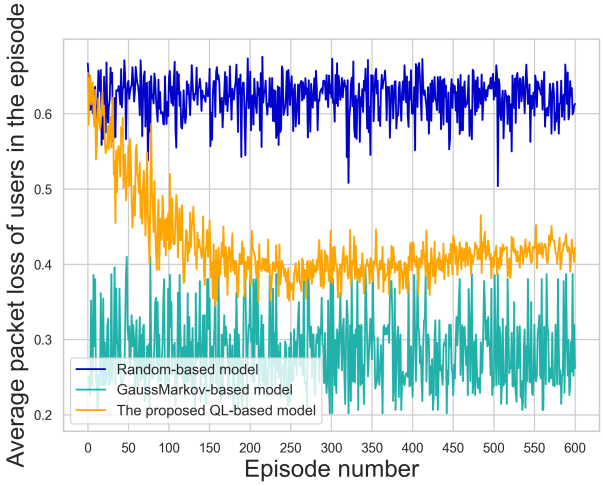
the same way as UEs move (which is one of the final aims of the RL-based agent), we can reach the sum data rate of around 43Mbps. Although the non-intelligent Gauss-Markov model faces many fluctuations in the performance results during the simulation, but it shows that the QL-based methodology has not maximized the sum data rate. We also compare the performance results of the proposed QL-based model with the two baseline in terms of users' average delay and users' average packet loss.



(a) Accumulated data rate of users in the proposed QL-based model vs. the two baselines



(b) Average transmission delay of users in the proposed QL-based model vs. the two baselines



(c) Average packet loss of users in the proposed QL-based model vs. the two baselines

Figure 3.5: Network performance comparison under the two baselines and the proposed QL-based UAV-BS placement

By comparing the results of the QL model and the random model, as illustrated in Fig 3.5b and Fig 3.5c, we can see that the QL-enabled UAV-BS algorithm gradually learns how to change its locations to decrease the average delay and the average packet loss of users starting from episode 1 to episode 600 where is converges after around 250 episodes. Similar to the Fig. 3.5a, the non-intelligent Gauss-Markov-based baseline has achieved lower delay and lower packet loss compared to the QL model although with many fluctuation.

As the aim of the QL algorithm is to maximize the cumulative reward the agent receives in a long run, we next visualize the cumulative reward the QL agent gets by performing its selected actions throughout the 600 episodes of training. Fig. 3.6 depicts this reward values which is calculated according to the Eq. 3.22 in each episode. As seen in this figure, the agent is able to step-by-step increase its received rewards throughout the training. The reward value starts from around -150 and it converges to around 50 after 250 episodes. However, the figure shows that there is a drop in the cumulative reward throughout the training, meaning that the UAV-BS gets stuck in local optimum. This is due to the fact that the state space of the proposed QL strategy is limited without taking into account the dynamic of users. This is the reason why in the next chapter we try to extend our state space to continuous.

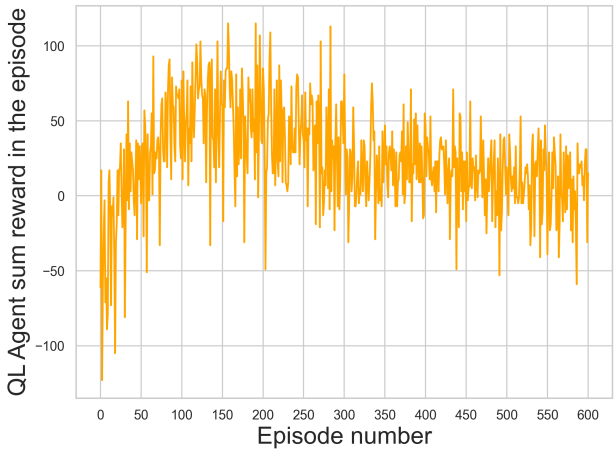


Figure 3.6: Accumulated reward of the agent in the proposed model

3.6 Conclusion

In this chapter, after proposing our system model, we have introduced a QL-based methodology in order to maximize the sum data rate of users. While the majority of the works in the literature assume that the ground users are fixed, this chapter has investigated the problem of UAV-BS deployment in the presence of mobile users. In our proposed QL strategy, the state space is discrete and the location information of users is not included in the state space. We have also designed a reward function that keeps the UAV-BS inside the boundaries of the investigated area while it also tries to maximize the sum data rate of the users. The proposed QL-based model has been compared with two baselines, one of which builds on a random mobility whereas the other one adopts a Gauss-Markov mobility model for the UAV-BSs. The simulation results have shown that the QL-based UAV-BS deployment model results in significantly better performance compared to the random deployment of UAV-BS which is outperformed by the Gauss-Markov-based UAV-BS deployment method. The reason is that the Gauss-Markov-based movement algorithm has a prior knowledge of how to move UAV-BS to match with users mobility. Due to the limitations of Q-table we have not been able to include the location information of users in the proposed model. This is the reason why in the next section we will present a DQL-based methodology in order to tackle this limitation of QL and involve the mobility information of users in the training process of the UAV-BS.

Chapter 4

Deep Q-learning-based deployment of UAV-BSs

4.1 Introduction

In the previous chapter, we proposed a QL-based methodology for the location optimization of UAV-BSs. The QL-based model used a Q-table which has to have a limited number of states as its rows, and actions as its columns. In order to restrict the number of states, we only considered the location information of the UAV-BS in our proposed state space, hence not involving the position of users in the training process. Moreover, in order to limit the number of actions that the UAV-BS can take, we defined our action space to involve five discrete actions corresponding to one unit towards or against the x and y axes plus no movement.
In this chapter, in order to tackle the limitations of Q-table, we propose a DQL-based strategy for the deployment of the UAV-BS in the air. In the proposed DQL model, the Q-table is replaced with a deep neural network (DNN) where the state of the environment is fed to the DNN as input and the Q-values associated with each pair of the input state and various actions are produced in the output of the DNN. By using a DNN instead of the Q-table, we, in fact, tackle the limitation about the number of allowed states since the DNN can receive an infinite number of states from a continuous state space as its input. Therefore, we can include the location information of users in addition to that of the UAV-BS in the state space of the DQL-based model. However, like in the QL model, the DQL-based method contains five discrete actions since the Q-values associated with each pair of state and action should be estimated by the DNN as the output. The proposed

reward function in chapter 3 is used for the DQL model as well where the DQL-enabled agent of the UAV-BS aims to maximize the reward values in a long run. Our simulation results show that our proposed DQL-based model outperforms the QL-based methodology that we proposed in Chapter 3. Moreover, the DQL method achieves the maximum data rate results that the Gauss-Markov-based baseline, introduced in Section 3, can reach. This means in our presented DQL-based strategy, the UAV-BS can autonomously adjust its locations according to the mobility of ground users and maintain its maximum performance even if the users move to different locations.

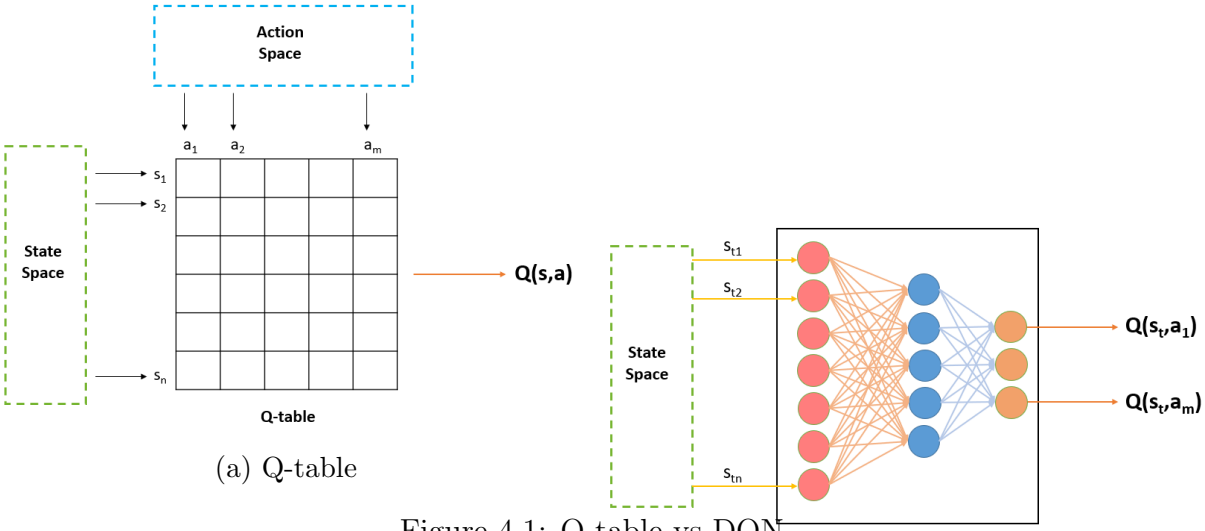
4.2 Preliminaries for deep Q-learning

QL is a promising solution which can efficiently obtain the optimal policy when the action and state spaces are small. However, in many real-world environments, the action and state spaces are very large due to which the amount of memory required to save and update Q-table would dramatically increase and the available storage would become insufficient [36]. Besides, the amount of time required to explore each state and obtain related Q-table would be significantly high. As a result, the QL algorithm may not be able to achieve an effective learning and find the optimal policy in large state-action environments [9].

An alternative approach to cope with the limitations of QL is DQL. In DQL, instead of having a Q-table which consists of all possible states and actions, a deep Q-network (DQN), which is a deep neural network (DNN) is used to estimate the Q-values, namely $Q(s, a)$ [105]. In fact, in DQN, the state is fed as the input, and the Q-values per action are obtained in the output. Figure 4.1 shows how Q-table and DQN differ.

In order to train the DQN and learn the Q function, the algorithm of DQL requires a training dataset with labels, also known as targets, to gradually learn the parameters, minimize the cost function and improve the predictions. In order to create training dataset and labels, DQL exploits two techniques called target network and experience replay (ER) which are explained as follows.

- **Target network**
  Looking into the TD equation expressed in Eq. 3.19, DQL considers $y = R(s, a) + \gamma \operatorname{argmax}_{a'} Q(s', a')$, containing the actual reward, as the actual target while the next predicted value of the DNN is considered as the Q-value [52]. Therefore, the system attempts to minimize the TD and make $Q_t(s, a)$ and $Q_{t+1}(s, a)$, expressed in Eq. 3.20, as close as possible to each other until they converge. In other words, the purpose of



(a) Q-table

(b) Deep Q-Network

Figure 4.1: Q-table vs DQN

DQL is to minimize the difference between y and Q using gradient descent and back propagation.

- **Experience replay**
  ER is used by DQL in order to build a training set as soon as the agent starts to explore the environment. ER stores all the transitions in the form of $(s_t, a_t, r_t, s'_{t+1})$ in a cyclic buffer, called replay memory; thus, the replay memory contains a set of previously observed data to be considered as the training set [52]. This mechanism allows the DQN to be trained more efficiently by using both old and new experiences and tackle the problem of having consecutive interdependent states that are alike. For example, when a self-driving car is driving in a straight line, if there is no ER in training phase, the network only considers a set of the current experience including consecutive similar transitions that converge very quickly without having a chance to learn the rest of the network [39]. However, with ER, the system can make use of both past experience and the latest experience simultaneously for training, and avoid an early wrong convergence.

The algorithmic flow of DQL is illustrated in Fig. 4.2 [90].

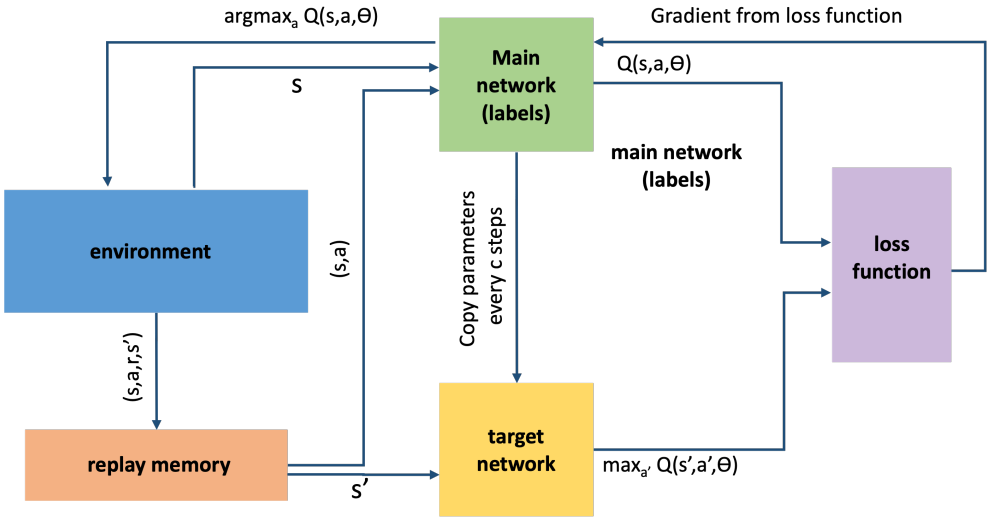


Figure 4.2: Algorithm flow of Deep Q-learning

4.3 The proposed DQL-based deployment of UAV-BSs

As mentioned, QL is applicable to the environments where the state and action spaces are limited. Otherwise, the memory needed to save Q-table and the time required to search the Q-table would significantly increase and make Q-learning ineffective under large state/action spaces. Therefore, in this section of the thesis, a DQL approach is introduced as an alternative RL-based approach to cope with the limitations of QL. Indeed, instead of employing QL as an intelligent algorithm of placing the UAV-BS, a DQL-based algorithm is adopted by considering the states of ground users as well as that of UAV-BS in the sky as parts of state space. In the proposed DQL-based solution, the Q-table is replaced with a DQN. At each time step t, the state information, containing the positions of users and UAV-BS, is fed to the DQN as input. The DQN gradually learns the optimal output (action) associated with each input state by trying to minimize the TD explained in Section 4.2. All the transitions in the form of $(s_t, a_t, r_t, s'_{t+1})$ are stored in the ER memory from which batch files of various old and new transitions are used to update the DQN parameters to learn the environment.

Indeed, in the proposed DQL, our aim is to minimize the TD loss(see Eq. 3.21) of the quality of performing a specific action a in a state s and make $Q_t(s,a)$ and $Q_{t+1}(s,a)$ as close as possible to each other until they converge. Fig. 4.3 illustrates the communication

between the intelligent agent of the UAV-BS in the proposed DQL-based methodology and the environment to learn the best actions associated with each state. The system model in the proposed DQL-based strategy builds on the model in Fig. 3.1 and Section 3.2.

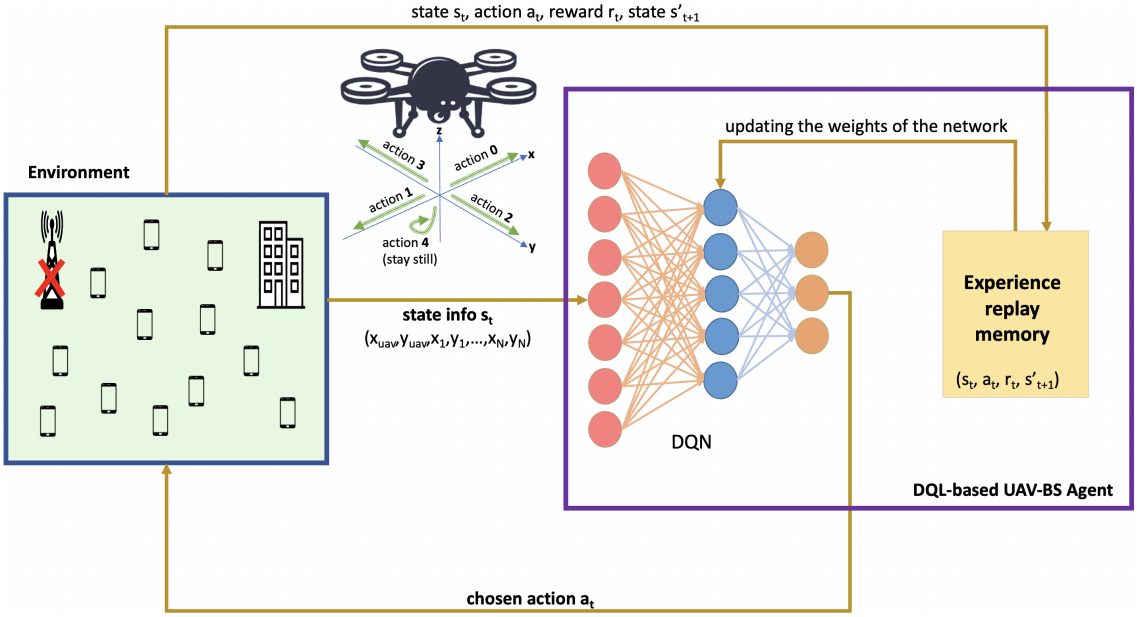


Figure 4.3: UAV-BS deployment with DQL

Unlike in the QL-based strategy, where we restrict the number of states to fit in the Q-table, the DQL-based methodology considers the user locations as part of the state information so that the model can adapt to user mobility more effectively. Based on this, the state space of our DQL-based deployment model is defined as follows:

State space: $(x_{\text{uav}}(t), y_{\text{uav}}(t), x_1(t), y_1(t), ..., x_n(t), y_n(t))$ where x_{uav} and y_{uav} are the x and y coordinates of the UAV-BS while x_i and y_i are the x and y coordinates of the i-th user. Different from the proposed QL-based strategy in Chapter 3, in the DQL model, the users and UAV-BS's coordinates can get continuous values since there is no Q-table with size limitation.

The action space and the reward function are defined the same as in the QL-based UAV-BS positioning methodology which we re-express as follows.

Action Space: The proposed action space from which the UAV-BS chooses action at each time step, consists of five different action codes, i.e. 0, 1, 2, 3, 4. These action codes

correspond to moving towards the positive direction of x axis, the negative direction of x axis, the positive direction of y axis, the negative direction of y axis and stay in the same coordinates with no movement, accordingly. The distance that UAV-BS moves at each time step towards or against the x and y axis depends on the speed of the UAV-BS. Thus, a UAV-BS at a speed of $g\,m/s$, moves g meters at each time interval if the chosen action is not 4.

Reward Function: As the objective is to maximize the data rates of users, the reward function in our model is formulated as follows where x_{uav} and y_{uav} are the x and y coordinates of the UAV-BS at time step t, and $R_i(t)$ is the achieved data rate of i-th user at time step t.

$$r_t = \begin{cases} -\beth & \text{if} \quad x_{\text{uav}}(t) \notin [0, x_{\max}] \text{ or } \ y_{\text{uav}}(t) \notin [0, y_{\max}] \\ 1 & \text{if} \quad \sum_{i=1}^{N} R_i(t) \text{ increases} \\ -1 & \text{if} \quad \sum_{i=1}^{N} R_i(t) \text{ decreases} \\ -0.2 & \text{if} \quad \sum_{i=1}^{N} R_i(t) \text{ does not change} \end{cases}, \tag{4.1}$$

The steps of the proposed DQL for 3D placement of an UAV-BS are summarized in Algorithm 2.

4.4 Simulation Results

Similar to the simulation setting in chapter 3, a square area of 1×1 km^2 with 50 users who are randomly distributed is selected for the simulation of our proposed framework in this chapter. The users follow Gauss-Markov mobility model with the speed between 2 m/s and 8 m/s. At each time step t, according to the proposed DQL-based strategy, the UAV-BS checks its current location and the current location of the ground users and predicts its best next movement among the five possible actions of the action space. Other environmental parameters of our simulation are selected according to the Table 3.2. Regarding the simulation of the DQN in our proposed DQL-based strategy, we consider 102 neurons in the input layer of the DQN which correspond to the x an y coordinates of the 50 UEs and the UAV-BS, while five neurons corresponding to 5 actions are added to the output layer. Moreover, after fine tuning the DQN, below parameters were selected for the DQL model:

Algorithm 2 DQL-based optimal placement of UAV-BS

Initialize the DQN with random weights
Initialize the ER memory
$\epsilon = 1$
for each episode **do**
initialize users randomly
initialize UAV-BS in one of the 3D corners with altitude h
for each time t step in episode **do**
observe $s_t = (x_{\text{uav}}, y_{\text{uav}}, x_{n_1}, y_{n_1}, x_{n_2}, y_{n_2}, ..., x_{n_N}, y_{n_N})$

$$a_t = \begin{cases} \text{random number} & \text{with the probability of } \epsilon \\ argmax(DQN.predict(s_t)) & \text{with the probability of } 1-\epsilon \end{cases}$$

Move UAV-BS according to the chosen action a_t
observe state $s_{t+1} =$
$(x'_{\text{uav}}, y'_{\text{uav}}, x'_{n_1}, y'_{n_1}, x'_{n_2}, y'_{n_2}, ..., x'_{n_N}, y'_{n_N})$
for every user n **do**
calculate R_n
end for

$$r_t = \begin{cases} -\Re & \text{if} \quad x_b(t) \notin [0, x_{max}] \quad \text{or} \quad y_b(t) \notin [0, y_{max}] \\ 1 & \text{if} \quad \sum_{n=1}^{N} R_n(t) \text{ increases} \\ -1 & \text{if} \quad \sum_{n=1}^{N} R_n(t) \text{ decreases} \\ -0.2 & \text{if} \quad \sum_{n=1}^{N} R_n(t) \text{ does not change} \end{cases}$$

store (s_t, a_t, r_t, s_{t+1}) in replay memory
randomly sample a batch of C samples (s_i, a_i, r_i, s_{i+1}) from the ER memory
Perform a gradient descent on Loss =
$(r_i + \gamma \max \text{DQN.predict}(s_{i+1}))$ - DQN.predict(s_i)

if $\epsilon \geq$ epsilon_min **then** $\epsilon = \epsilon \times$ epsilon_decay

else $\epsilon =$ epsilon_min
end if
end for
end for

Two hidden layers, each of which having 50 neurons, are considered as the hidden layers of the DQN. The learning rate α is set to 0.01, while a discount factor γ of 0.95 is applied. The parameters related to the DQN are presented in Table 4.1. These parameters are

initially selected according to [84], and they have been slightly fine-tuned for our proposed methodology. Moreover, the significantly negative reward $\mathfrak{R}$ that is given to the agent if it goes outside of the boundaries is set to -5 which is big enough compared to the negative reward of -1 the agent receives if the data rate of users drops by its movement. We run our simulation for 600 episodes where each episode starts with a different distribution of users. Each episode of the simulation consists of 200 time steps ($T = 200s$) where at each time step the intelligent agent of the UAV-BS perceives the location information of the users and that of the UAV-BS, and decides about the best movement of the UAV-BS so as to maximize the users' sum data rate.

In order to compare the results of our proposed DQL-based methodology with baselines, we consider an AI-based QL strategy that we proposed in Chapter 3 plus the two conventional algorithms for moving the UAV-BS, i.e. random movement of the UAV-BS and the Gauss-Markov-based movement of the UAV-BS. Therefore, the baselines for the validation of our presented DQL model are defined as follows.

1. **Random Action:** For this baseline, at each time step, the UAV-BS randomly selects its next movement from the five possible actions. So, the UAV-BS tries to cover the entire area of interest by randomly moving in the sky.

2. **Gauss-Markov-based mobility model:** In this baseline, the mobility model of the UAV-BS and users are the same. In fact, by defining this baseline, our aim is

Table 4.1: DQN parameters of the simulation in the proposed DQL-based methodology (selected and fine-tuned based on [84])

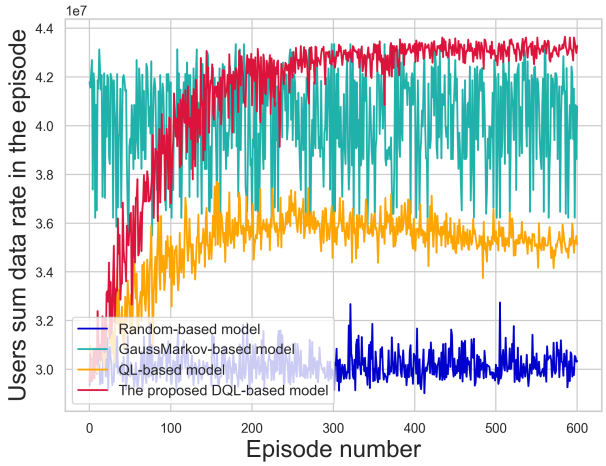
Parameter	**Value**
learning rate	0.01
discount factor a	0.95
epsilon decay b	0.995
minimum epsilon	0.1
replay memory buffer size	100000
batch size	128
episodes	600
time steps in each episode	200

to see how the performance of the network improves if users and the UAV-BS follow identical mobility models with the same parameters such as direction and speed.

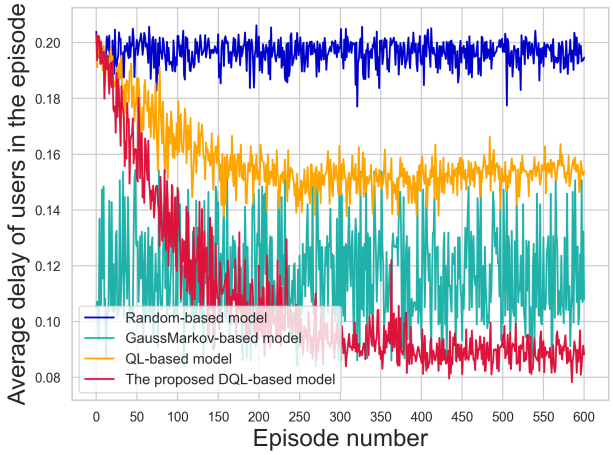
3. **QL-based mobility model** In this baseline, at each time step, the UAV-BS selects its best movement according to the proposed QL-based strategy in Chapter 3. The state space in this baseline is discrete and does not include the location of users, and the Q values associated with each pair of states and actions are stored in a Q-table.

We first validate the performance of our proposed DQL-based methodology in terms of users sum data rate as maximizing the data rate of users is the main objective of this solution. As illustrated in Fig. 4.4a, the DQL-based strategy has significantly outperformed the baselines with regards to the users' sum data rate. As depicted, the QL-based model, wherein the state space is discrete and the location information of users at each time step is not included in the state representation of the environment, is not able to increase the users' data rate more than 36Mbps. However, the proposed DQL-based method that adds the location of users as part of the state space has successfully achieved the sum data rate of more than 43Mbps which is a great performance improvement compared to the QL-based model. By looking at the sum data rate of users in Gauss-Markov-based baseline in Fig. 4.4a, where the users and the UAV-BS follow the same mobility, we realize that the DQL-based strategy has reached the maximum data rate we can achieve by moving the UAV-BS in the direction of users. The DQL-based model even surpasses the Gauss-Markov model by converging to the maximum data rate while the Gauss-Markov faces many fluctuation during the simulation.

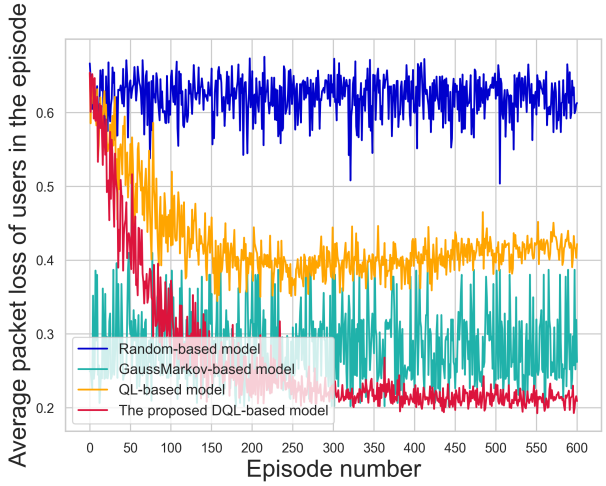
We also compare the results of the proposed DQL-based model with regards to the transmission delay and packet loss of users to see how these two metrics improve if the UAV-BS's moves intelligently in the sky with the use of proposed DQL algorithm. Fig. 4.4b and 4.4c shows how the users average delay and average packet loss improve throughout the 600 episodes of training under the proposed DQL-based strategy compared to the three baselines. Regarding the users average delay, the proposed DQL-based model has reduced the delay to 0.08s while the QL-based strategy has converged to the transmission delay of 0.15s after 250 episodes. Similarly, the proposed DQL-based algorithm has minimized the users' average packet loss ratio to 0.2 while the QL-based method converges to 0.4 for the users' average packet loss.



(a) Accumulated data rate of users in the proposed DQL-based model vs. the three baselines



(b) Average transmission delay of users in the proposed DQL-based model vs. the three baselines



(c) Average packet loss of users in the proposed DQL-based model vs. the three baselines

Figure 4.4: Network performance comparison under the three baselines and the proposed DQL-based UAV-BS placement

Furthermore, we compare the cumulative reward the agent receives over an episode in the proposed DQL-based model and the QL-based baseline. As Fig. 4.5 shows, the DQL-based method surpasses the QL-based baseline in terms of the agent's received rewards. The sum of the rewards the DQL-based agent of the UAV-BS receives is around 200 after 600 episodes while the QL-based solution converges to around 0. This proves that the target of the RL-based algorithm which is defined as maximizing the cumulative reward of the agent over a long run, is much more effectively met in the DQL-based solution as opposed to the QL-based methodology.

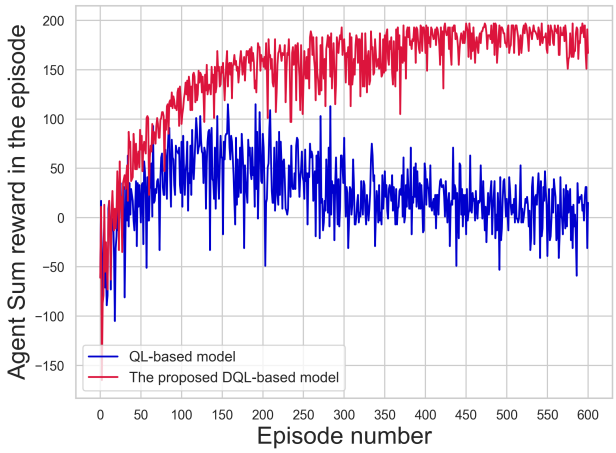


Figure 4.5: Comparison of the agent reward under the QL-based baseline and the proposed DQL-based UAV-BS placement

We also illustrate the DQN's loss value of the proposed DQL-based solution in Fig. 4.6. As mentioned earlier, in DQL, a DQN tries to minimize the TD loss using back propagation and gradient descent until the Q values of various states and actions converge. Fig. 4.6 depicts how this loss value decreases over the 600 episodes where it converges. This decrease in TD loss value and its convergence is another evidence that the DQL algorithm has successfully learned how to minimize the TD loss and predict the Q-values properly.

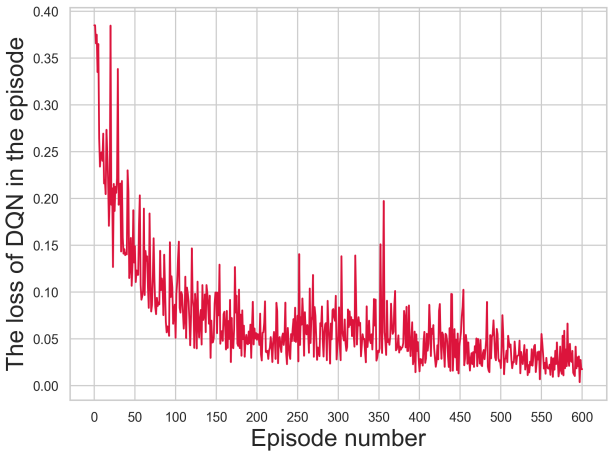


Figure 4.6: The DQN's loss in the proposed DQL-based UAV-BS placement

4.5 Conclusion

In this chapter of the thesis, we have proposed a DQL-based methodology for the placement and movement of the UAV-BS in the sky with the aim of maximizing the users sum data rate. While in the previous chapter we limited the number of states by not including the location of users in the state representation of the environment, in this chapter, we have presented a DQL-based algorithm in order to add the location information of users to the state space. In fact, we have replaced the Q-table of the QL algorithm, presented in Chapter 3, with a DQN in this chapter to overcome the limitation of QL that cannot include unlimited number of states. The proposed DQN learns to predict the correct Q-values associated with each pair of state and action by minimizing the TD loss. Our simulation results have shown that the DQL-based method can notably increase the users sum data rate compared to the QL-based solution and the other two conventional baselines. With regards to the other two performance metrics, i.e. users average transmission delay and users' average packet loss also, the proposed DQL model has outperformed all the baselines. Therefore, the DQL-based algorithm, wherein the location of users and UAV-BS are both fed to the DQN to predict the best action at each time step, can much more effectively deploy the UAV-BS in the air and increase the network performance. Similar to the QL algorithm in the previous chapter, the DQL-based solution in this chapter uses a discrete action space that contains five actions for moving the UAV-BS at each time step. To extend the action space to continuous, we introduce continuous actor-critic RL, a more advanced RL-based algorithm for UAV-BS placement, in the next chapter.

Chapter 5

Actor-critic deep Q-learning-based deployment of UAV-BSs

5.1 Introduction

In Chapter 3, we deployed a QL-based model for the placement and movement of the UAV-BS in the air with the objective of maximizing the users sum data rate in a UAV-assisted wireless network. While the state and action spaces were both discrete in the proposed QL-based method, in Chapter 4, we presented a DQL-based UAV-BS deployment strategy to extend the state space to continuous so that the location information of users can be added to the state space. Extending the state space from discrete to continuous notably improved our performance results. For further improving the network performance, in this chapter, we expand the action space from discrete to continuous by proposing a continuous actor-critic deep Q-learning (ACDQL) methodology for the deployment of UAV-BS in the air. In the proposed ACDQL-based model, at each time step, the UAV-BS can choose its next action from a continuous range $(-k, k)$. The value of k is the maximum distance the UAV-BS can move and it depends on the maximum speed of the UAV-BS. In fact, QL and DQL are both value-based RL algorithms where the Q-values associated with each pair of state and action should be calculated at each time step. Therefore, it is infeasible to have an unlimited number of actions. The solution we propose to tackle this limitation and have unlimited number of actions, is a continuous ACDQL algorithm that exploits a policy-based function to estimate the best action from a continuous action space. The simulation results of this chapter show that the proposed ACDQL-based solution outperforms all the baselines, including QL and DQL-based models, in terms of users' sum data rate, users'

packet loss and users' transmission delay. Moreover, the proposed ACDQL-based method converges after around 30 episodes while the QL and DQL models converge after more than 200 episodes which seems a significant improvement for a UAV-assisted wireless network.

5.2 Preliminaries for actor-critic deep Q-learning

DQL is a powerful type of RL which is suitable for the environments that have large state and action spaces [22] where Q-table is unable to efficiently store and search among a large number of states and actions. However, there is one limitation for DQL: it requires discrete action space, otherwise the training could be computationally very intensive and not feasible to get converged [27]. Actor-critic deep Q-learning (ACDQL) methods, as a solution to this limitation, can handle continuous action space by employing a policy function in addition to the value function which is used in QL and DQL algorithms as well [102].

As mentioned earlier in this thesis, in order to maximize the sum data rate of users, at each time step t, the UAV-BS needs to perceive the state of the environment and decide what movement is the best to take. Since deciding about the next location of the UAV-BS depends on the current state of the environment, the problem of solving the 3D location optimization of UAV-BS can be expressed as MDP explained in Section 3.3 which is solvable by RL algorithms. MDP consists of four main elements (S, A, π, R) that should be defined for an environment so that it can get solved with RL. As stated earlier, S is the state space containing all possible states of the environment, A is the action space consisting of all the actions the RL agent can take, and R stands for the reward function. π represents the policy function which is the mapping from the states to a probability distribution over various actions. The aim of RL is to find the optimal policy, π_{opt}, that maps each state to the action that leads to maximizing the long-term reward. RL algorithms are divided into three groups where each group has a different approach for finding the optimal policy. These three groups are described below.

- **Critic-only methods:** QL and DQL are both among the critic-only methods that have no explicit function for the policy π. They, instead, use a state-action value function Q for estimating the optimal policy. In fact, the optimal policy π_{opt} is approximated by performing an optimization strategy over the value function $Q(s, a)$ (see Eq. 3.18), which is expressed as

$$\pi_{\text{opt}}(a) = \underset{a}{\operatorname{argmax}}\, Q(s, a) \tag{5.1}$$

This value function optimization is approximated by minimizing the TD loss like what we performed in our proposed QL and DQL-based strategies, by making use of a Q-table and a DQN, in Chapter 3 and 4, respectively. In both QL and DQL, $Q(s, a)$ should be calculated for all possible actions in a state before the algorithm decides which action is the best to be taken in that state. This is the reason why none of QL and DQL can support continuous action space.

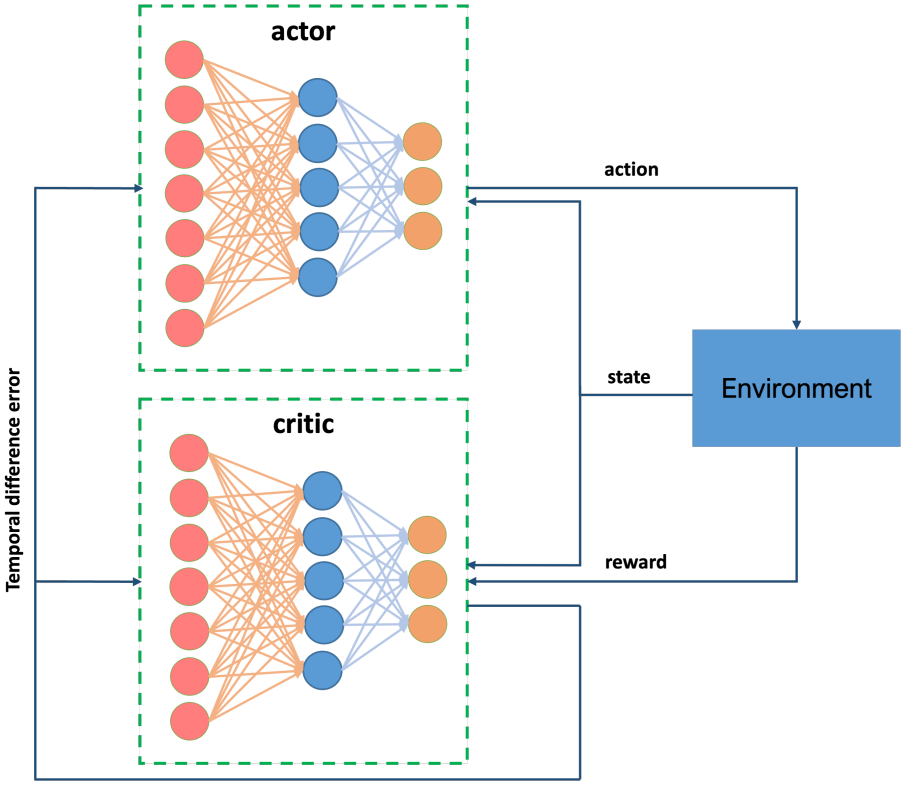


Figure 5.1: Actor-critic deep Q-learning

- **Actor-only methods:** Actor-only methods uses a policy gradient approach to directly work with the policy to reach the optimal policy that maximizes the expected reward over the time period T. The objective function can be formulated as [27]

$$J(\pi_\theta) = E_{\pi_\theta}[\sum_{t=0}^{T-1} R(s_t, a_t)] \tag{5.2}$$

where $E(R(s_t, a_t))$ is the expected long-term reward by performing action a in state

s. The objective function J should be maximized by adjusting the policy parameter θ that always maximize the reward. In order to update the policy parameter θ which results in the maximized objective function, the gradient ascent rule is used. unlike the critic-only methods, such as QL and DQL, actor-only methods allow the policy to generate actions in the continuous action space. However, due to the highly variable behavior of dynamic environments, actor-only methods suffer from the high variance in the estimated policy gradient that results in a very slow learning [27].

- **Actor-critic methods:** Actor-critic algorithms consist of two parts, i.e. actor and critic. The actor makes use of the policy gradient method to produce continuous actions while the critic uses TD loss technique to evaluate the actions selected by the actor [62]. In fact, the role of the critic is to tackle the large variance in the policy gradient that exists in actor-only methods. Critic implements and optimizes $Q(s, a)$ by using TD loss over the last experiences in order to assess the effectiveness of the selected actions by the actor. After selecting an action in a specific state, the critic network criticizes the selected action which leads to modifying the policy so that the actions with higher Q-values will be more frequently selected in the future and vice versa.

ACDQL is a type of actor-critic methods where the actor and critic are implemented by deep neural networks. Figure 5.1 is a high-level representation of actor and critic networks in ACDQL. In this chapter, we propose an ACDQL-based strategy that makes use of actor and critic networks for optimizing the deployment of UAV-BS in the air where the UAV-BS can choose its movements from continuous action space. The details of this strategy are explained in the next section.

5.3 The proposed ACDQL-based deployment algorithm

In this section of the thesis, our final strategy for improving the placement optimization of the UAV-BS is presented that uses an ACDQL-based approach with an extended action space from discrete to continuous. In our proposed continuous ACDQL-based methodology, the four elements of the MDP, i.e. states, actions, rewards, and policy, denoted by (S, A, π, R), are introduced as follows.

State space: Similar to the DQL-based model which was proposed in Chapter 4, the state space in our proposed ACDQL-based methodology is formulated as the location of the UAV-BS plus the locations of all ground users at each time step t which is written as

$$\mathbf{s}_t = (x_{\text{uav}}(t), y_{\text{uav}}(t), x_1(t), y_1(t), ..., x_n(t), y_n(t))$$

Action space: The action space of our continuous ACDQL-based model is defined as the displacement of the UAV-BS at each time step t over the x and y axes which are denoted by $d^x_{\text{uav}}(t)$ and $d^y_{\text{uav}}(t)$, respectively. The UAV-BS's displacement along the x and y coordinates at each time step can be any value in the continuous range of $[-k, k]$ where k is the maximum speed of the UAV-BS. So, the action space of our proposed methodology is expressed as

$$\mathbf{a}_t = (d^x_{\text{uav}}(t), d^y_{\text{uav}}(t))$$

Reward function: Since the objective of our proposed ACDQL-based model is to maximize the sum data rate of users over the time period T, any movement of the UAV-BS that increases the users' sum data rate at time t, compared to that at time $t-1$, receives an immediate positive reward. In contrast, the agent is penalized if its movement at time t leads to smaller sum data rate of users compared to time $t-1$. Therefore, our reward function is presented as

$$r_t = \begin{cases} -\mathfrak{R} & \text{if} \quad x_{\text{uav}}(t) \notin [0, x_{\max}] \text{ or} \\ & \quad y_{\text{uav}}(t) \notin [0, y_{\max}] \\ \sum_{i=1}^{N} R_i(t) - \sum_{i=1}^{N} R_i(t-1) & \text{otherwise} \end{cases}, \tag{5.3}$$

where $\mathfrak{R}$ is a big negative reward the agent receives if it moves outside the boundaries and it is defined for the UAV-BS agent to comply with the boundary constraint of our proposed optimization problem in Eq. 3.10. It should be noted that, the proposed ACDQL strategy uses a continuous action space where the displacement of the UAV-BS can be any value in the range of $[-k, k]$ and our aim is to learn the best value of this displacement at each time step. Therefore, we need to differentiate between the reward that the agent receives by selecting various values of this displacement. With that being said, we cannot use the same reward function we proposed in the QL and DQL-based strategies as the received reward of the agent was not variable there.

Policy function: The policy function in our proposed model should return the long-term cumulative rewards the UAV-BS agent receives by performing action a in state s. By deriving the policy function, the agent, at each time step, can choose the action that

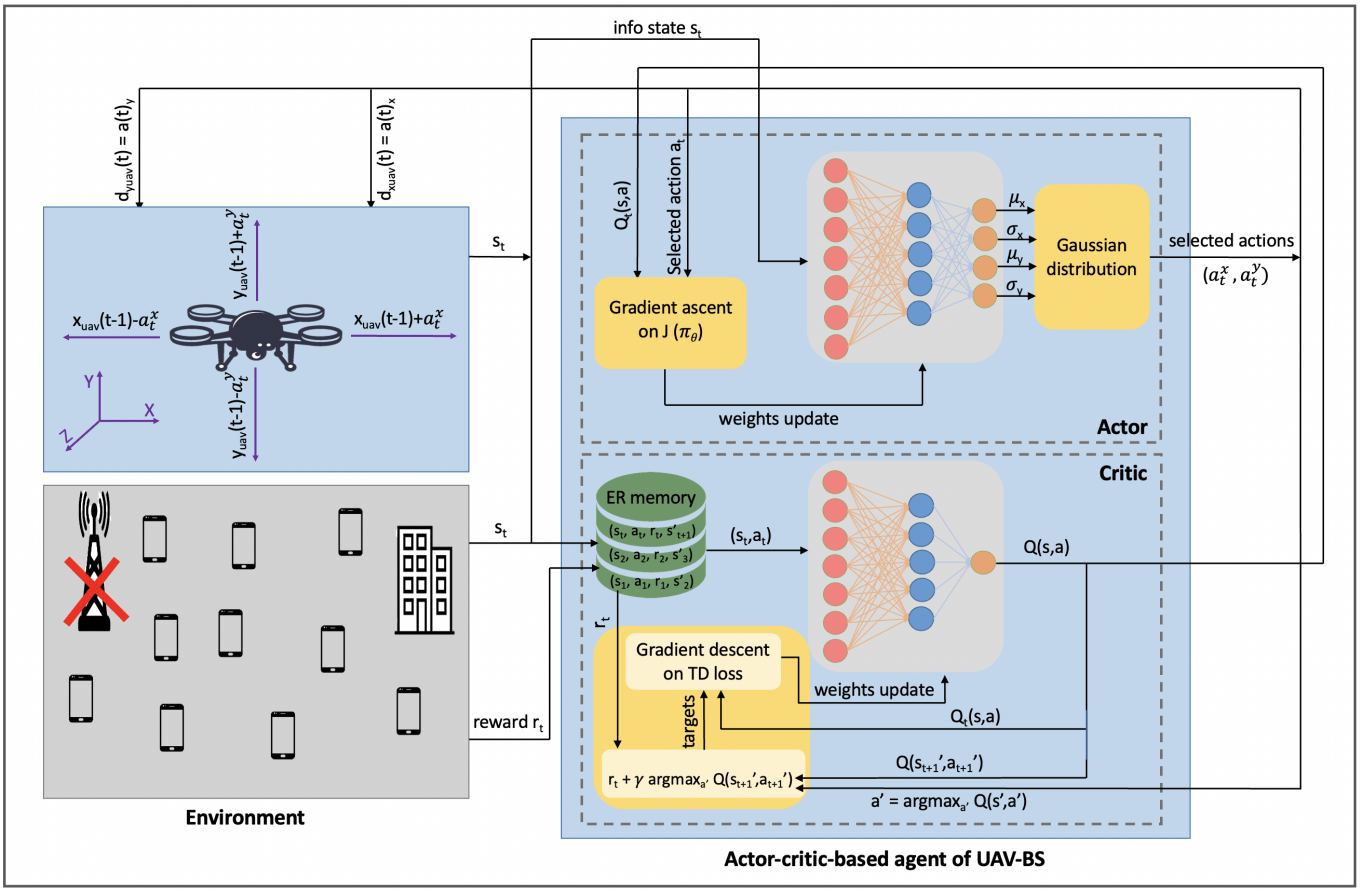


Figure 5.2: Proposed continuous actor-critic-based UAV-BS's deployment

would result in maximizing the long-term cumulative rewards and consequently maximize the users' sum data rate. Therefore, in order to solve the location optimization of the UAV-BS, we only need to implement a strategy to discover the optimal policy function.

Our proposed strategy for deriving the optimal policy function is based on a continuous ACDQL algorithm that consists of two deep neural networks (DNNs), i.e. an actor DNN and a critic DNN. The actor network receives the state of the environment as its input and maps it to a probability distribution $\pi_{\theta^a}(s)$, where θ^a is the weights of the actor's DNN, to predict what action is the best to be chosen. In other words, the actor network represents our required policy function. The critic network, on the other hand, implements a value function in the form of $Q(s, a)$ that evaluates the quality of the action a chosen by the

actor network in state s. Fig. 5.2 is an illustration of our proposed ACDQL-based UAV-BS deployment algorithm. The system model again follows what proposed in Section 3.2 and Fig. 3.1.

More details of the actor and critic networks in our proposed solution are explained as follows.

Proposed actor network: The aim of the actor network is to obtain the optimal policy function π_{θ^a} that maximizes the long-term expected rewards. Thus, a policy gradient method is used in the actor network to generate the optimal policy. The policy gradient on the actor's DNN can be calculated by [76]

$$\nabla_{\theta^a} J(\pi_{\theta^a}) = E[\sum_{t=0}^{t-1} \nabla_{\theta^a} log(\, \pi_{\theta^a}(a_t|s_t))\, Q(s_t, a_t)] \tag{5.4}$$

where $Q(s_t, a_t)$ is the quality of performing action a_t in state s_t which is calculated by the critic network.

Since our proposed UAV-BS's deployment algorithm is designed for continuous action space, the actor network uses Gaussian distribution, which is a well-known multi-parameter distribution for a continuous random variables, in order to select the action. More specifically, our proposed actor network outputs the mean (μ) and standard deviation (σ) of the Gaussian distribution of each input state for selecting the optimal action. Thus, the parameterized policy according to the Gaussian distribution can be formulated as [88]

$$\pi_{\theta^a}(a_t|s_t) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(a-\mu(s))^2}{2\sigma^2}) \tag{5.5}$$

The standard deviation of the Gaussian distribution in the actor network is usually considered as a fixed value in the literature. However, in this thesis, we propose to generate the σ as one of the outputs of the actor's DNN so that the agent can dynamically balance between exploration and exploitation. By considering σ as one of the outputs of the actor's DNN, the agent can manage to explore more with larger values of σ, while more exploitation happens with the smaller values of σ [19]. The value of σ gets close to zero with the convergence of the UAV-BS's deployment algorithm.

Finally, the actor network uses the gradient ascent technique on the policy gradient, shown in Eq. 5.4, in order to update its parameters and gradually converge to the optimal policy. So, the actor network iteratively updates θ^a which is written as

$$\nabla_{\theta^a} = \nabla_{\theta^a} + \alpha^a log(\, \pi_{\theta^a}(a_t|s_t))\, Q(s_t, a_t), \tag{5.6}$$

where α^a is the learning rate of the actor's DNN.

Proposed critic network: The critic network which is supposed to validate the quality of the action selected by the actor network, makes use of the Bellman equation to evaluate the quality of performing action a in state s based on the expected discounted reward which is formulated as [86]

$$Q(s, a) = r(s, a) + \gamma \operatorname*{argmax}_{a'} Q(s', a'), \tag{5.7}$$

where s' is the destination state after performing action a in state s, and γ is the discount factor that quantifies the importance of the future rewards.

Considering our random environment and that the value of $Q(s, a)$ changes over time, temporal difference (TD) is introduced in the critic network to represent the difference between the new predicted $Q_t(s, a)$ and the old $Q_{t-1}(s, a)$. TD can be expressed as [52]

$$TD(s, a) = r(s, a) + \gamma \operatorname*{argmax}_{a'} Q(s', a') - Q_{t-1}(s, a). \tag{5.8}$$

By applying TD in Eq. 5.7, the value function Q is transformed to

$$\begin{aligned} Q_t(s, a) &= Q_{t-1}(s, a) + \alpha^c(r(s, a) \\ &+ \gamma \operatorname*{argmax}_{a'} Q(s', a') - Q_{t-1}(s, a)), \end{aligned} \tag{5.9}$$

where α^c denotes the learning rate of the critic's DNN.

Looking into Eq. 5.9, the DNN of the critic network tries to minimize the TD by considering $y = R(s, a) + \gamma \operatorname{argmax}_{a'} Q(s', a')$ as the network's true labels, also know as targets, while $Q_t(s, a)$ is counted as the predicted value of the network. Therefore, a gradient descent technique is utilized in the critic network in order to minimize the TD loss which is written as

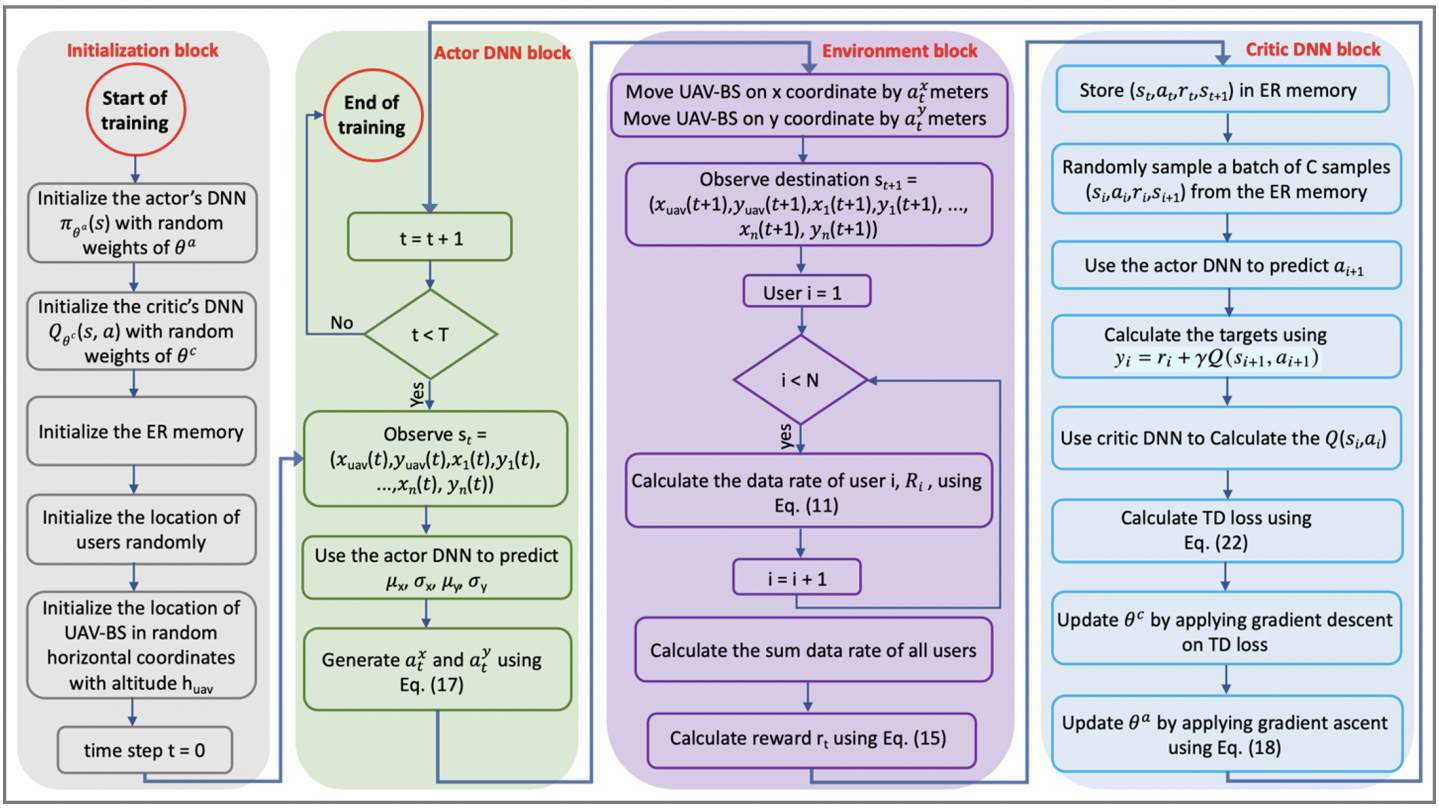


Figure 5.3: The flowchart of the proposed ACDQL-based algorithm for the optimal placement of UAV-BS

$$L(\theta^c) = E[(y_t - Q_t(s, a)|\theta^c)^2], \tag{5.10}$$

where θ^c is the weights of the critic's DNN.

The calculated TD loss is then used to update θ^c. It should be noted that, the action value that satisfies $\text{argmax}_{a'} Q(s', a')$ in Eq. 5.10 is determined by the actor network.

Moreover, an ER memory (see Section 4.2) is used in the proposed critic network to store all the agent's experience in the form of $(s_t, a_t, r_t, s'_{t+1})$. With utilizing the ER memory, the critic network will be able to choose random batches of both old and new experiences for training the DNN and minimizing the TD loss more effectively [52].

Flowchart 5.3 is the steps of our proposed continuous ACDQL-based strategy.

5.4 Simulation Results

The Simulation setting is the same as what we presented in Section 3.5. The area of interest, where the users are located and move, is 1×1 km^2. 50 users are randomly distributed and move according to the Gauss-Markov mobility model and their speed is between 2 m/s and 8 m/s. At each time step t, the proposed continuous ACDQL-based algorithm decides about the distance the UAV-BS should move along the x and y coordinates. The maximum speed of the UAV-BS is 6 m/s. Therefore, the maximum distance the ACDQL algorithm can select at each time step is 6 m along or against the x and y coordinates. Table 3.2 lists the other environmental parameters of our simulation.
The actor's DNN consists of an input layer of 102 neurons, corresponding to the x and y coordinates of the UAV-BS and users at each time step, two hidden layers and an output layer. The two hidden layers are fully-connected and each one has 256 neurons while the output layer consists of four neurons corresponding to μ_x, σ_x, μ_y and σ_y. The activation function for the hidden layers of the actor's network is Relu function while Tanh activation function is selected for the output layer. The Adam optimizer is used as the optimization algorithm of the actor's DNN with the learning rate of $\alpha^a = 0.0001$.
Similar to the actor network, the critic's DNN consists of an input layer of 102 neurons, where the state information of the UAV-BS and users is received, and two fully-connected hidden layers that each one has 256 neurons. Since the critic network evaluates the quality of action a in state s, the first hidden layer of the critic's DNN concatenate the state received from the input layer and the action taken from the actor network and pass it through the next layers. The output layer of the critics' DNN consists of a single neuron for outputting $Q(s, a)$. Relu activation function is utilized in the hidden layers of the critic network while the linear combination of the outputs in the last hidden layer forms the Q-value of the output layer with no activation function in the output layer. The Adam optimizer with the learning rate of $\alpha^c = 003$ is adopted in the critic network. Table 5.1 lists the hyper parameters of the actor and critic DNNs in our simulation which are initially selected according to [11] and then slightly fine-tuned for our proposed ACDQL-based strategy.

The simulation is run for 600 episodes where each episode consists of 200 time steps ($T = 200s$) and the initial position of users change at the beginning of each episode. At each time step, the ACDQL-based algorithm selects what distance the UAV-BS should move that will maximize the data rate of users.

We compare the performance result of our proposed ACDQL-based methodology with four baselines described below:

Table 5.1: DQN parameters of the simulation in the proposed ACDQL-based methodology

Parameter	**Value**
actor's learning rate α^a	0.0001
critic's learning rate α^c	0.003
discount factor γ	0.99
replay memory buffer size	1000000
batch size	128
episode count	600
time steps in each episode	200

1. **Random model**: In this baseline model, at each time step t, the UAV-BS randomly chooses a direction and a distance between $[-k, k]$ to move.

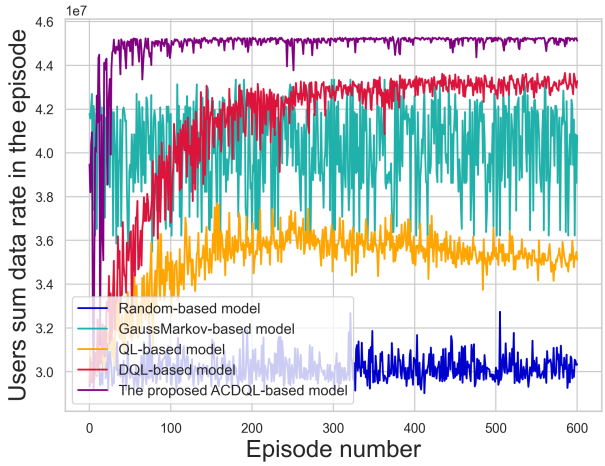
2. **Gauss-Markov model**: As the second baseline for the movement of the UAV-BS, we use the same mobility model that we employ for the movement of users with the same speed, direction, etc. By selecting this baseline, we aim to see the network performance if the movement pattern of the UAV-BS is matched with that of users.

3. **QL-based model**: In this model, the UAV-BS is equipped with a QL-based algorithm, proposed in Chapter 3, to intelligently select the best movement of the UAV-BS at each time step according to the QL strategy. A Q-table is used in this model for storing the Q-values where the rows represent the states and columns are the possible actions. The state and action space are both discrete and limited in this model in order to have a practical and feasible Q-table. Taking into account the maximum speed of the UAV-BS which is set to 6 m/s, the action space consists of 5 discrete actions for moving the UAV-BS six meters towards or against the x and y axis at each time step.

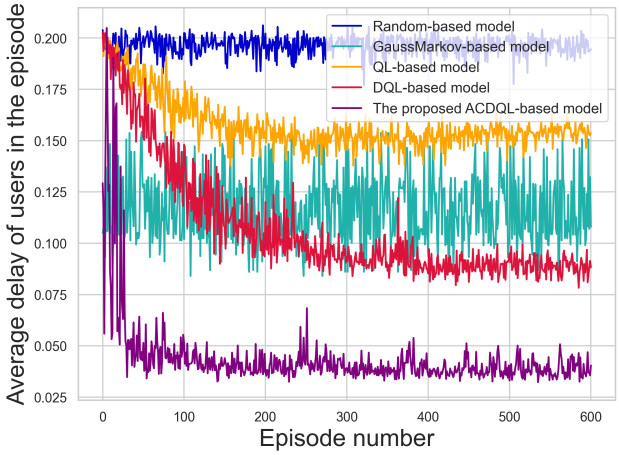
4. **DQL-based model**: As the fourth baseline, a DQL-based strategy, proposed in Chapter 4, is adopted for the movement of the UAV-BS at each time step. In this baseline model, a DQN is exploited to map the state and action pairs to Q-values. Since the Q-table is replaced with a DQN, in this baseline, the state space is continuous. The action space, however, is discrete and includes 5 actions similar to the QL-based model.

Since the objective of our proposed model is to move the UAV-BS towards the locations that maximize the users sum data rate, we first compare the performance result of the proposed model with the four baselines in terms of users' sum data rate. As depicted in Fig. 5.4a, the proposed ACDQL-based methodology achieves a significantly higher sum data rate compared to all the other baselines. More specifically, the ACDQL-based model maximizes the users' sum data to more than 45Mbps. This is while the proposed DQL-based solution can not increase the data rate to more than 43Mbps, and the QL-based methodology converges to 35Mbps. Moreover, the ACDQL-based method converges much faster than the two other intelligent solutions, i.e. QL and DQL-based models. The convergence for the proposed ACDQL-based model occurs after around 30 episodes while the QL and DQL-based baselines converge after around 200 episodes.

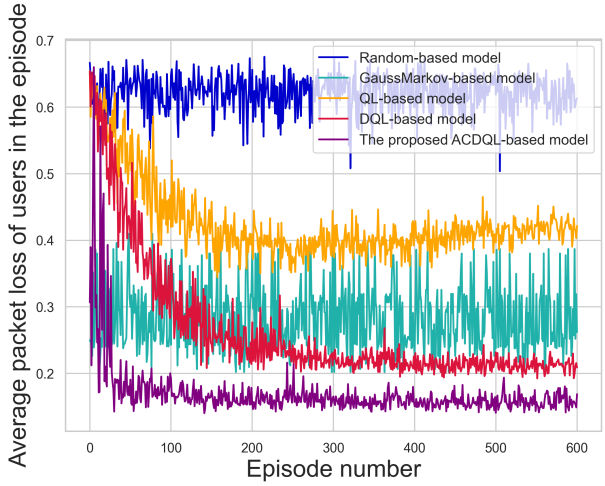
In addition, we evaluate the performance of our proposed ACDQL model in terms of users' average transmission delay and users' average packet loss. As illustrated in Fig. 5.4b and Fig. 5.4c, the proposed ACDQL-based model outperforms the four baselines with regards to both transmission delay and packet loss. More precisely, according to Fig. 5.4b, moving the UAV-BS in the sky according to the proposed ACDQL-based solution, minimizes the users average delay to 0.04s while the DQL and QL-based strategies can decrease the delay to 0.08s and 0.15s, respectively. Similarly, according to Fig. 5.4c, the ACDQL algorithm optimizes the movement of the UAV-BS to reach the average users packet loss ratio of 0.15, while the DQL and QL-based baselines can only decrease the average packet loss to 0.2 and 0.4, respectively. Therefore, the continuous ACDQL-based model has been able to autonomously move the UAV-BS towards the locations that not only maximizes the users' data rate, which is the main objective of the network, but also minimizes the transmission delay and packet loss where the agent converges in less than 30 episodes which is a significant improvement compared to QL and DQL strategies.



(a) Accumulated data rate of users in the proposed ACDQL-based model vs. the four baselines



(b) Average transmission delay of users in the proposed ACDQL-based model vs. the four baselines



(c) Average packet loss of users in the proposed ACDQL-based model vs. the four baselines

Figure 5.4: Network performance comparison under the four baselines and the proposed ACDQL-based UAV-BS placement

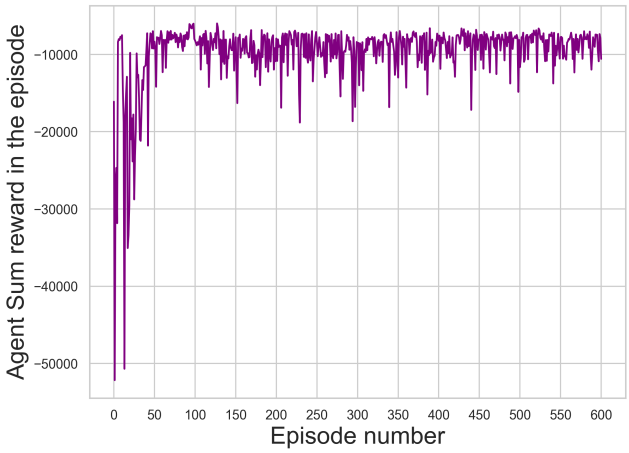


Figure 5.5: Accumulated reward of the agent in the proposed ACDQL-based model

As mentioned earlier, the main objective of any RL algorithm, including ACDQL, is to maximize the cumulative reward the agent receives over a long run. We formulated the reward function of the ACDQL-based method in Eq. 5.3 where the agent receives positive reward if its last action improves the users' data rate and vice versa. If the agent moves outside the boundaries, the agent receives a negative reward of $\mathfrak{R} = -1000$. Fig. 5.5 illustrates how the accumulated rewards the agent receives in each episode of the training gradually increases until it converges.

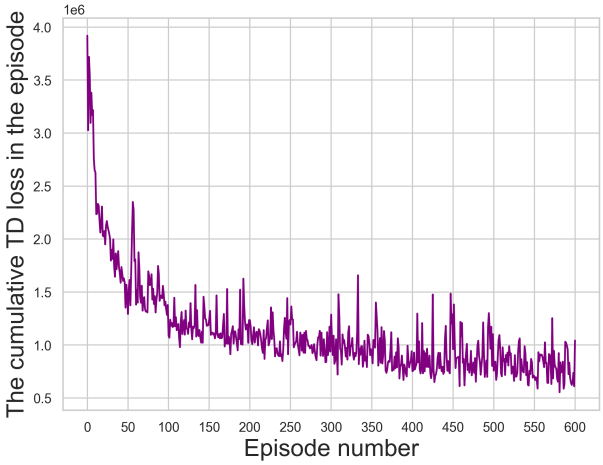


Figure 5.6: Accumulated TD loss of the agent in the proposed ACDQL-based model

Regarding the loss value of the DQN in the proposed solution, as expressed in Eq. 5.10, the critic's network tries to minimize the TD loss by using the gradient descent technique until the optimized policy function is derived and the Q values are converged. The decrease

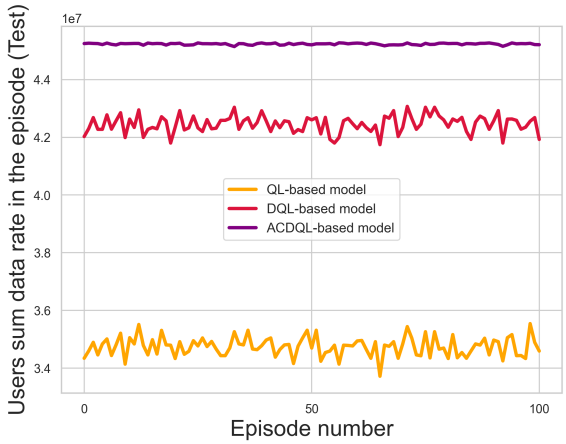


Figure 5.7: Users sum data rate comparison under the proposed QL, DQL, and ACDQL-based strategies (Test Phase)

in TD loss is illustrated in Fig. 5.6 which is another indicator of how the agent succeeds in effectively calculating the Q-values and estimating the best actions in various states.

Finally, we compare the test's performance result of our ACDQL-based methodology with the other two RL-based solutions, i.e. QL and DQL-based baselines. In order to do so, we store the trained models of ACDQL, QL and DQL after 600 episodes of training. The stored models are then used for 100 new episodes where each episode has a different user distribution which differs from those available in the training phase. The saved models determine the distance and direction the UAV-BS should move at each time step with no further change in the saved models. Fig. 5.7 compares the accumulated users' data rate over 100 episodes of test. As depicted in the picture, the proposed continuous ACDQL-based solution surpasses QL and DQL-based models wherein a discrete action space is used.

5.5 Conclusion

In this chapter of the thesis, we proposed our last strategy, i.e. a continuous ACDQL-based strategy, for the location optimization of the UAV-BS. While the action space in the previous proposed QL and DQL-based methodologies in Chapters 3 and 4 was discrete, in this chapter, we extended our action space to continuous. In fact, QL and DQL are both critic-only RL algorithms that are infeasible of having a continuous action space. Therefore, in the proposed QL and DQL-based strategies, at each time step, the UAV-BS could only choose from a few predefined actions to move. In order to overcome the limitation in the number of allowed movements, in this chapter, we proposed to use a continuous ACDQL-based method. ACDQL is an actor-critic RL method which consists of two networks, i.e. actor network and critic network. The actor network is capable of generating actions in a continuous action space while the critic network evaluates the action selected by the actor network. Our simulation results have shown that the proposed continuous ACDQL-based solution outperforms the previous solutions, i.e. QL and DQL-based methods in terms of users' sum data rate, users' average delay and users' average packet loss. Moreover, the ACDQL-based model converges after only 30 episodes of training while the convergence of the QL and DQL-based solutions occurs after more than 200 episodes.

Chapter 6

Conclusion

Uncrewed aerial vehicle-mounted base stations (UAV-BSs) or widely known as drone base stations, have recently gained increasing attention as a solution to provide Internet connectivity to users. They can be deployed to support terrestrial BSs during an occasional crowded event, in the case of a terrestrial BS failure, or during natural disasters. Furthermore, they can be positioned in the sky of remote areas, where deploying terrestrial BSs is not feasible, to provide the region with wireless Internet connection. Despite their tremendous benefits, UAV-BSs pose some critical challenges, such as channel modeling, optimal 3D placement, limited flight time, etc. In this thesis, we have focused on one of the most important challenges of UAV-BSs, which is the 3D location optimization of UAV-BSs. Solving the placement optimization problem of UAV-BSs has been addressed by various researchers by utilizing either heuristics methods or recently by ML-based solutions which are more suitable for dynamic networks. Among all type of ML algorithms, reinforcement learning (RL), specifically Q-learning (QL) and deep Q-learning (DQL), are considered as the most suitable methods for finding the best location of UAV-BSs and their optimal movement in the air, which have been the focus of many studies in recent years. However these studies have two limitations: 1-the users are located in fixed positions and their mobility is not considered while the UAV-BSs' location optimization algorithms are developed or analyzed. 2-the action space of the QL/DQL algorithms are discrete which means that the UAV-BS can only choose from a limited number of actions to move at each time step.

Therefore, in this thesis, we have extended the state-of-the-art and improved the placement optimization of UAV-BSs by considering the mobility of users while developing our RL-based solutions. We have also extended the action space of RL from discrete to continuous so that the UAV-BS will not be limited to a few actions to move, but instead, it

can move by any distance. In order to do so, we have first considered the Gauss-Markov mobility model for users who are mobile in this thesis. Then, we have proposed our reward function, which is used by RL algorithm, in a way that the ground users sum data rate would be maximized. Afterward, in order to build our UAV-BS deployment algorithm in the presence of mobile users, we have started by proposing a QL strategy wherein a Q-table is used and the state and action spaces are both discrete and the aim is to maximize the cumulative received rewards. Next, we have improved the UAV-BS deployment strategy by replacing the Q-table with a deep Q-network and proposing a DQL strategy wherein the state space is extended from discrete to continuous. Further improvement in the UAV-enabled wireless network has been done by proposing our last strategy, i.e. continuous actor-critic deep Q-learning (ACDQL)-based deployment of UAV-BS. In the continuous ACDQL-based solution, the action space, from which the UAV-BS can choose its next movement at each time step, has been extended from discrete to continuous. Our performance results have shown that the last proposed strategy, which is based on continuous ACDQL, significantly outperforms the proposed QL and DQL solutions as well as the conventional baselines, in terms of users data rate, transmission delay and packet loss. In terms of users sum data rate, the QL-based solution has reached around 36Mbps during 600 episodes of training. The DQL-based model has increased the sum data rate to around 43Mbps while the ACDQL has successfully converged to around 45Mbps. Regarding the average delay of users, the continuous ACDQL has minimized the delay to 0.03s while the DQL and QL-based strategies have reached 0.08s and 0.15s, respectively. With regards to the users packet loss, the continuous ACDQL algorithm has successfully minimized the packet loss ratio to 0.14, while the DQL and QL-based methods have reached 0.21 and 0.41, respectively. Another notable improvement of the ACDQL-based methodology compared to the QL and DQL-based solutions is the convergence time of the ACDQL algorithm. Thus, the proposed ACDQL-based methodology reduces the convergence time of the UAV-BS placement optimization by 85 percent compared to the Q-learning and deep Q-learning baselines which is a remarkable improvement

6.1 Future Directions

Our suggested improvement directions on this thesis are described as follows.

- **Energy-aware deployment of UAV-BSs** As mentioned in Section 1.2, UAV-BSs use rechargeable batteries that bring about limited operation time for them. In this thesis, we have not considred this energy limitation when proposing our UAV-BSs deployment algorithms. Considering that the energy limitation of the UAV-BSs is a serious constraint in practical scenarios, deploying an energy-aware ACDQL-based methodology for the placement optimization of UAV-BSs is a possible direction for the future studies. In order to do so, the optimization problem we presented in Section 3.2.2, needs to be extended as a joint optimization problem of data rate maximization and energy consumption minimization. An energy consumption model will be investigated and the state and action spaces will be extended accordingly.

- **Extending the UAV-assisted wireless network from a single UAV-BS to multiple UAV-BSs:** In many practical scenarios, for example during the temporary overcrowded events, a single UAV-BS might be insufficient for covering all the users. In such scenarios, deploying multiple UAV-BSs in the air is required to fulfill the Internet demand of users. Bringing multiple UAV-BSs to the scene, however, needs a collision avoidance system to be deployed so that the UAV-BSs do not collide. Therefore, extending this thesis from a single UAV-BS to multiple UAV-BSs would be another direction for the future studies. In order to follow this direction, the reward function needs to be extended in order to consider the distance between UAV-BSs. In addition, the state and action spaces should be extended so that the placement of multiple UAV-BSs can be jointly optimized.

- **Vision-based deployment of UAV-BSs:** In this thesis, we have assumed that the location information of the users is available to the UAV-BS and can be used by the intelligent agent inside the UAV-BS for the location optimization. However, in some of the practical scenarios, the position information of users might be unavailable to UAV-BSs. Therefore, deploying a convolutional neural network (CNN)-based algorithm that enable UAV-BSs so sense their environment would be a promising future direction. By deploying a CNN-based algorithm on the UAV-BS and equipping the UAV with cameras and other necessary hardware, the UAV-BS can capture the position of ground users, as well as all the obstacles and other UAV-BSs. The output of the CNN algorithm, which is a representation of the environment, can then be fed as the input state to the ACDQL-based algorithm to decide about the next movement of the UAV-BS.

References

[1] Softbank demonstrates drone wireless relay system for disaster recovery and locating persons trapped under debris. Technical report, SoftBank News, Sep. 2020.

[2] Shubhani Aggarwal and Neeraj Kumar. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, 14, October 2019.

[3] Akram Al-Hourani, Sithamparanathan Kandeepan, and Abbas Jamalipour. Modeling air-to-ground path loss for low altitude platforms in urban environments. In *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, pages 2898–2904. IEEE, 2014.

[4] Akram Al-Hourani, Sithamparanathan Kandeepan, and Simon Lardner. Optimal lap altitude for maximum coverage. *IEEE Wireless Communications Letters*, 3(6), July 2014.

[5] Fadi Al-Turjman, Joel Poncha Lemayian, Sinem Alturjman, and Leonardo Mostarda. Enhanced deployment strategy for the 5g drone-bs using artificial intelligence. *IEEE Access*, 7, June 2019.

[6] Sher Ali, Amir Haider, Muhibur Rahman, Muhammad Sohail, and Yousaf Bin Zikria. Deep learning (dl) based joint resource allocation and rrh association in 5g-multi-tier networks. *IEEE Access*, 9:118357–118366, 2021.

[7] Eduardo Nuno Almeida, Kelwin Fernandes, Francisco Andradey, Pedro Silvay, Rui Campos, and Manuel Ricardo. A machine learning based quality of service estimator for aerial wireless networks. In *2019 International Conference on Wireless and Mobile Computing, Networking and Communication*, pages 1–6. IEEE, 2019.

[8] Mohamed Alzenad, Amr El-Keyi, Faraj Lagum, and Halim Yanikomeroglu. 3-d placement of an unmanned aerial vehicle base station (uav-bs) for energy-efficient maximal coverage. *IEEE Wireless Communications Letters*, 6(4), August 2017.

[9] Kai Arulkumaran andMarc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning. *IEEE Signal Processing Magazine*, 34(6), November 2017.

[10] G. Ausiello, P. Crescenzi, and M. Protasi. Approximate solution of np optimization problems. *Theoretical Computer Science*, 150(1), October 1995.

[11] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Reinforcement learning through asynchronous advantage actor-critic on a gpu. In *International Conference on Learning Representations*, May 2017.

[12] Harald Bayerlein, Paul De Kerret, and David Gesbert. Trajectory optimization for autonomous flying base station via reinforcement learning. In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications*, pages 1–5. IEEE, Jun. 2018.

[13] Hande Y. Benson. Mixed integer nonlinear programming using interior-point methods. *Optimization Methods Softw.*, 26(6):911–931, April 2010.

[14] R. Irem Bor-Yaliniz, Amr El-Keyi, and Halim Yanikomeroglu. Efficient 3-D Placement of an Aerial Base Station in Next Generation Cellular Networks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2016.

[15] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5), September 2002.

[16] Oktay Cetinkaya, Domenico Balsamo, and Geoff V Merrett. Internet of mimo things: Uav-assisted wireless-powered networks for future smart cities. *IEEE Internet of Things Magazine*, 3(1):8–13, 2020.

[17] Sathyanarayanan Chandrasekharan, Karina Gomez, Akram Al-Hourani, Sithamparanathan Kandeepan, Tinku Rasheed, Leonardo Goratti, David Grace Laurent Reynaud, Isabelle Bucaille, Thomas Wirth, and Sandy Allsopp. Designing and implementing future aerial communication networks. *IEEE Commun. Mag.*, 54(5), May 2016.

[18] Cihan Tugrul Cicek, Hakan Gultekin, Bulent Tavli, and Halim Yanikomeroglu. Uav base station location optimization for next generation wireless networks: Overview and future research directions. In *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*. IEEE, 2019.

[19] Thomas Degris, Patrick M. Pilarski, and Richard S. Sutton. Model-free reinforcement learning with continuous action in practice. In *2012 American Control Conference (ACC)*. IEEE, June 2012.

[20] Reza Zanjirani Farahani, Nasrin Asgari, Nooshin Heidari, Mahtab Hosseininia, and Mark Goh. Covering problems in facility location: A review. *Elsevier Computers and Industrial Engineering*, 62(1), February 2012.

[21] Anders Forsgren. Inertia-controlling factorizations for optimization algorithms. *Applied Numerical Mathematics*, 43(2):91–107, October 2002.

[22] Boris Galkin, Erika Fonseca, Ramy Amer, Luiz Dasilva, and Ivana Dusparic. Reqiba: Regression and deep q-learning for intelligent uav cellular user to base station association. *IEEE Transactions on Vehicular Technology*, pages 1–1, 2021.

[23] Boris Galkin, Jacek Kibilda, and Luiz A. DaSilva. Deployment of uav-mounted access points according to spatial user locations in two-tier cellular networks. In *2016 Wireless Days*, pages 1–6. IEEE, 2016.

[24] Boris Galkin, Jacek Kibilda, and Luiz A. DaSilva. Uavs as mobile infrastructure: Addressing battery lifetime. *IEEE Communications Magazine*, 57(6):132–137, February 2019.

[25] Qingji Gao, Bingrong Hong, Zhendong He, Jie Liu, and Guochen Niu. An improved q-learning algorithm based on exploration region expansion strategy. In *2006 6th World Congress on Intelligent Control and Automation*. IEEE, Jun 2006.

[26] Rozhina Ghanavi, Elham Kalantari, Maryam Sabbaghian, Halim Yanikomeroglu, and Abbas Yongacoglu. Efficient 3D aerial base station placement considering users mobility by reinforcement learning. In *2018 IEEE Wireless Communications and Networking Conference*, pages 1–6. IEEE, 2018.

[27] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(6), Dec. 2012.

[28] Jiangchun Gu, Guoru Ding, Yitao Xu, Haichao Wang, and Qi hui Wu. Proactive optimization of transmission power and 3D trajectory in uav-assisted relay systems with mobile ground users. *Chinese Journal of Aeronautics*, 34(3):16, 2020.

[29] Jianli Guo, Yonghua Huo, Xiujuan Shi, Jiahui Wu, Peng Yu, Lei Feng, and Wenjin Li. 3D aerial vehicle base station (uav-bs) position planning based on deep q-learning for capacity enhancement of users with different qos requirements. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management*, pages 1508–1512. IEEE, 2019.

[30] Lav Gupta, Raj Jain, and Gabor Vaszkun. Survey of important issues in uav communication networks. *IEEE Communications Surveys and Tutorials*, 18(2), April 2016.

[31] Samira Hayat, Evşen Yanmaz, and Raheeb Muzaffar. Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint. *IEEE Communications Surveys and Tutorials*, 18(4), April 2016.

[32] Aicha Idriss Hentati and Lamia Chaari Fourati. Comprehensive survey of uavs communication networks. *Computer Standards and Interfaces*, 72, 2020.

[33] Jaroslav Holis and Pavel Pechac. Elevation dependent shadowing model for mobile communications via high altitude platforms in built-up areas. *IEEE Transactions on Antennas and Propagation*, 56(4), April 2008.

[34] Yiming Huo, Xiaodai Dong, Tao Lu, Wei Xu, and Marvin Yuen. Distributed and multilayer uav networks for next-generation wireless communication and power transfer: A feasibility study. *IEEE Internet of Things Journal*, 6(4), August 2019.

[35] Md Zahidul Islam, Vladimir Estivill-Castro Md Anisur Rahman, and Terry Bossomaier. Combining k-means and a genetic algorithm through a novel arrangement of genetic operators for high quality clustering. *Expert Systems With Applications*, 91:402–417, January 2018.

[36] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, 7, Sep 2019.

[37] Elham Kalantari, Muhammad Zeeshan Shakir, Halim Yanikomeroglu, and Abbas Yongacoglu. Backhaul-aware robust 3D drone placement in 5G+ wireless networks. In *2017 IEEE International Conference on Commun. Workshops, Networking and Commun.*, pages 109–114. IEEE, 2017.

[38] Elham Kalantari, Halim Yanikomeroglu, and Abbas Yongacoglu. On the number and 3D placement of drone base stations in wireless cellular networks. In *2016 IEEE 84th Vehicular Technology Conference*, pages 1–6. IEEE, 2016.

[39] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, Feb 2021.

[40] Paulo V. Klaine, Joao P. B. Nadas, Richard D. Souza, and Muhammad A. Imran. Distributed drone base station positioning for emergency cellular networks using reinforcement learning. *cognitive computation*, 10, May 2018.

[41] Rolf Klein. Abstract voronoi diagrams and their applications. In *Computational Geometry and its Applications*, pages 148–157. Springer, 1988.

[42] Marek Kulbacki, Jakub Segen, Wojciech Knieć, Ryszard Klempous, Konrad Kluwak, Jan Nikodem, Julita Kulbacka, and Andrea Serester. Survey of drones for agriculture automation from planting to harvest. In *2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES)*, pages 000353–000358. IEEE, 2018.

[43] Dave Lee. Facebook abandons its project aquila flying internet plan. Technical report, 2018.

[44] Jon Lee and Sven Leyffer. *Mixed Integer Nonlinear Programming*. The IMA Volumes in Mathematics and its Applications. Springer, 2012.

[45] Bin Li, Zesong Fei, and Yan Zhang. Uav communications for 5g and beyond: Recent advances and future trends. *IEEE Internet of Things Journal*, 6(2):2241–2263, April 2019.

[46] Bin Li, Zesong Fei, Yan Zhang, and Mohsen Guizani. Secure uav communication networks over 5g. *IEEE Wireless Communications*, 26(5), July 2019.

[47] Peiming Li and Jie Xu. Placement optimization for uav-enabled wireless networks with multi-hop backhauls. *Journal of Communications and Information Networks*, 3(4), December 2018.

[48] Xiaohui Li and Andrey V Savkin. Networked unmanned aerial vehicles for surveillance and monitoring: A survey. *Future Internet*, 13(7):174, 2021.

[49] Chi Harold Liu, Xiaoxin Ma, Xudong Gao, and Jian Tang. Distributed energy-efficient multi-uav navigation for long-term communication coverage by deep reinforcement learning. *IEEE Transactions on Mobile Computing*, 19(6), June 2020.

[50] Xiao Liu, Yuanwei Liu, and Yue Chen. Deployment and movement for multiple aerial base stations by reinforcement learning. In *2018 IEEE Globecom Workshops*, pages 1–6. IEEE, 2018.

[51] Xiao Liu, Yuanwei Liu, and Yue Chen. Reinforcement learning in multiple-uav networks: Deployment and movement design. *IEEE Transactions on Vehicular Technology*, 68(8), August 2019.

[52] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking- a survey. *IEEE Communications Surveys and Tutorials*, 21(4), 2019.

[53] Jiangbin Lyu, Yong Zeng, Rui Zhang, and Teng Joon Lim. Placement optimization of uav-mounted mobile base stations. *Computer Networks*, 21(3), March 2017.

[54] Yael Maguire and Kevin Waclawicz. Aquila: What's next for high-altitude connectivity? Technical report, Facebook, 2017.

[55] Wendel Melo, Marcia Fampa, and Fernanda Raupp. An overview of minlp algorithms and their implementation in muriqui optimizer. *Annals of Operations Research*, 286(6):217–241, March 2020.

[56] Balmukund Mishra, Deepak Garg, Pratik Narang, and Vipul Mishra. Drone-surveillance for search and rescue in natural disaster. *Computer Communications*, 156:1–10, 2020.

[57] Farhan Mohammed, Ahmed Idries, Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar. Uavs for smart cities: Opportunities and challenges. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, May 2014.

[58] Mehrdad Moradi, Karthikeyan Sundaresan, and Eugene Chai. Skycore: Moving core to the edge for untethered and reliable uav-based lte networks. *Computer Standards and Interfaces*, 72, 2018.

[59] Naser Hossein Motlagh, Tarik Taleb, and Osama Arouk. Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives. *IEEE Internet of Things*, 3(6), September 2016.

[60] Mohammad Mozaffari, Ali Taleb Zadeh Kasgari, Walid Saad, Mehdi Bennis, and Merouane Debbah. 3d cellular network architecture with drones for beyond 5g. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, December 2018.

[61] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, Young-Han Nam, and Mérouane Debbah. A tutorial on uavs for wireless networks: Applications, challenges, and open problems. *IEEE Communications Surveys and Tutorials*, 21(3), March 2019.

[62] Jelle Munk, Jens Kober, and Robert Babuška. Learning state representation for deep actor-critic control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016.

[63] Kamesh Namuduri, Yan Wan, and Mahadevan Gomathisankaran. Mobile ad hoc networks in the sky: state of the art, opportunities, and challenges. In *Proceedings of the second ACM MobiHoc workshop on Airborne networks and communications*. ACM, Jul. 2013.

[64] Ram Gopal Lakshmi Narayanan and Oliver C.Ibe. Chapter 6 - joint network for disaster relief and search and rescue network operations. In *Wireless Public Safety Networks 1 Overview and Challenges*, pages 163 – 193. Elsevier, 2015.

[65] Deepshikha Pandey and Punit Pandey. Approximate q-learning: An introduction. In *International Conference on Machine Learning and Computing*, pages 317–320. IEEE, 2010.

[66] Art Pregler. When cows fly: At&t sending lte signals from drones. Technical report, AT&T Technology Blog, 2017.

[67] Wang Qiang and TZhan Zhongli. Reinforcement learning model,algorithms and its application. In *International Conference on Mechatronic Science, Electric Engineering and Computer*, pages 1143–1148. IEEE, 2011.

[68] Ali Raza, Syed Hashim Raza Bukhari, Farhan Aadil, and Zeshan Iqbal. An uav-assisted vanet architecture for intelligent transportation system in smart cities. *International Journal of Distributed Sensor Networks*, 17(7):15501477211031750, 2021.

[69] Nadisanka Rupasinghe, Yavuz Yapıcı, İsmail Güvenç, Monisha Ghosh, and Yuichi Kakishima. Angle feedback for noma transmission in mmwave drone networks. *IEEE Journal of Selected Topics in Signal Processing*, 13(3), Mar. 2019.

[70] Jon Russell. Facebook is reportedly testing solar-powered internet drones again — this time with airbus. Technical report, 2019.

[71] Andrey V. Savkin and Hailong Huang. Deployment of unmanned aerial vehicle base stations for optimal quality of coverage. *IEEE wireless communication letters*, 8(1), February 2019.

[72] Hazim Shakhatreh, Ahmad H. Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *IEEE Access*, 7, April 2019.

[73] Vishal Sharma, Mehdi Bennis, and Rajesh Kumar. Uav-assisted heterogeneous networks for capacity enhancement. *IEEE Commun. Lett.*, 20(6), April 2016.

[74] Weisen Shi, Junling Li, Wenchao Xu, Haibo Zhou, Ning Zhang, Shan Zhang, and Xuemin Shen. Multiple drone-cell deployment analyses and optimization in drone assisted radio access networks. *IEEE Access*, 6, February 2018.

[75] Joohyun Shin, Thomas A. Badgwell, Kuang-Hung Liu, and Jay H. Lee. Reinforcement learning – overview of recent progress and implications for process control. *Computers and Chemical Engineering*, 127, August 2019.

[76] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*. ACM, June 2014.

[77] Isaac J. Sledge and José C. Príncipe. Balancing exploration and exploitation in reinforcement learning using a value of information criterion. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Mar 2017.

[78] Zaib Ullah, Fadi Al-Turjman, Leonardo Mostarda, and Roberto Gagliardi. Applications of artificial intelligence and machine learning in smart cities. *Computer Communications*, 154:313–323, March 2020.

[79] L.S. Vargas, V.H. Quintana, and A. Vannelli. A tutorial description of an interior point method and its applications to security-constrained economic dispatch. *EEE Transactions on Power Systems*, 8(3):1315–1324, August 1993.

[80] Manfred Dutch von Ehrenfried. Chapter 5 - commercial corporations and applications. In *Stratospheric Balloons*. Springer, Cham, 2021.

[81] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: an overview. *Soft Computing*, 22:387–408, 2018.

[82] Haijun Wang, Haitao Zhao, Weiyu Wu, Jun Xiong, Dongtang Ma, and Jibo Wei. Deployment algorithms of flying base stations: 5g and beyond with uavs. *IEEE Internet of Things Journal*, 6(6), December 2019.

[83] Li-Chun Wang, Chuan-Chi Lai, Hong-Han Shuai, Hsin-Piao Lin, Chi-Yu Li, Teng-Hu Cheng, and Chiun-Hsun Chen. Communications and networking technologies for intelligent drone cruisers. In *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, December 2019.

[84] Qiang Wang, Wenqi Zhang, Yuanwei Liu, and Ying Liu. Multi-uav dynamic wireless networking with deep reinforcement learning. *IEEE Communications Letters*, 23(12), December 2019.

[85] Zhe Wang, Lingjie Duan, and Rui Zhang. Adaptive deployment for uav-aided communication networks. *IEEE Transactions on Wireless Communications*, 18(9):4531–4543, September 2019.

[86] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3), May 1992.

[87] Adam C. Watts, Vincent G. Ambrosia, and Everett A. Hinkley. Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use. *Remote Sensing*, 4(6), April 2012.

[88] Yifei Wei, F. Richard Yu, Mei Song, and Zhu Han. User scheduling and resource allocation in hetnets with hybrid energy supply: An actor-critic reinforcement learning approach. *IEEE Transactions on Wireless Communications*, 17(1), Jan 2018.

[89] Marco Wiering and Martijn van Otterlo. *Reinforcement Learning-State-of-the-Art*. Springer, 2012.

[90] Jiahui Wu, Peng Yu, Lei Feng, Fanqin Zhou, Wenjing Li, and Xuesong Qiu. 3D aerial base station position planning based on deep q-network for capacity enhancement. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management*, pages 482–487. IEEE, 2019.

[91] Xing Xi, Xianbin Cao, Peng Yang, Jingxuan Chen, Tony Quek, and Dapeng Wu. Joint user association and uav location optimization for uav-aided communications. *IEEE Wireless Communications Letters*, 8(6), August 2019.

[92] Peng Yang, Xianbin Cao, Chao Yin, Zhenyu Xiao, Xing Xi, and Dapeng Wu. Proactive drone-cell deployment: Overload relief for a cellular network under flash crowd traffic. *IEEE Transactions on Intelligent Transportation Systems*, 18(10), October 2017.

[93] Wei Yu, Hansong Xu, James Nguyen, Erik Blasch, Amirshahram Hematian, and Weichao Gao. Survey of public safety communications: User-side and network-side solutions and future directions. *IEEE Access*, 6, November 2018.

[94] Yong Zeng, Xiaoli Xu, and Rui Zhang. Trajectory design for completion time minimization in uav-enabled multicasting. *IEEE Transactions on Wireless Communications*, 17, April 2018.

[95] Yong Zeng and Rui Zhang. Energy-efficient uav communication with trajectory optimization. *IEEE Transactions on Wireless Communications*, 16(6):3747–3760, March 2017.

[96] Yongs Zeng, Qingqing Wu, and Rui Zhang. Accessing from the sky: A tutorial on uav communications for 5g and beyond. *Proceedings of the IEEE*, 107(12), December 2019.

[97] Chiya Zhang, Weizheng Zhang, Wei Wang, Lu Yang, and Wei Zhang g. Research challenges and opportunities of uav millimeter-wave communications. *IEEE Wireless Communications*, 26(1), February 2019.

[98] Qianqian Zhang, Mohammad Mozaffari, Walid Saad, Mehdi Bennis, and Merouane Debbah. Machine learning for predictive on-demand deployment of uavs for wireless communications. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, December 2018.

[99] Xiaogang Zhang and Zhijing Liu. An optimized q-learning algorithm based on the thinking of tabu search. In *2008 International Symposium on Computational Intelligence and Design*. IEEE, Oct 2008.

[100] Haitao Zhao, Haijun Wang, Weiyu Wu, and Jibo Wei. Deployment algorithms for uav airborne networks toward on-demand coverage. *IEEE Journal on Selected Areas in Communications*, 36(9), August 2018.

[101] Yi Zheng, Yuwen Wang, and Fanji Meng. Modeling and simulation of pathloss and fading for air-ground link of haps within a network simulator. In *2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 421–426. IEEE, 2013.

[102] Chen Zhong, Ziyang Lu, M. Cenk Gursoy, and Senem Velipasalar. A deep actor-critic reinforcement learning framework for dynamic multichannel access. *IEEE Transactions on Cognitive Communications and Networking*, 5(4):1125–1139, 2019.

[103] Xukai Zhong, Yiming Huo, Xiaodai Dong, and Zhonghua Liang. Deep q-network based dynamic movement strategy in a uav-assisted network. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*. IEEE, Dec. 2020.

[104] Yi Zhong, Tony Q. S. Quek, and Xiaohu Ge. Heterogeneous cellular networks with spatio-temporal traffic: Delay analysis and scheduling. *IEEE Journal on Selected Areas in Communications*, 35(6), March 2017.

[105] Hao Zhu, Yang Cao, Wei Wang, Tao Jiang, and Shi Jin. Deep reinforcement learning for mobile edge caching: Review, new features, and open issues. *IEEE Network*, 32(6), Nov 2018.