Machine Learning Methods for UAV-aided Wireless Networks

Harald Bayerlein

Machine Learning Methods for UAV-aided Wireless Networks

Dissertation

submitted to

Sorbonne Université

in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Author:

Harald Bayerlein

Publicly defended on the 7th of September 2021, before a committee composed of:

Reviewers

Prof.	**Mehdi Bennis**	University of Oulu, Finland
Prof.	**Andrea M. Tonello**	University of Klagenfurt, Austria

Examiners

Prof.	**Roberto Verdone**	University of Bologna, Italy
Prof.	**Sofie Pollin**	KU Leuven, Belgium
Prof.	**Florian Kaltenberger**	EURECOM, France

Thesis Advisor

Prof.	**David Gesbert**	EURECOM, France

Méthodes d'apprentissage automatique pour l'utilisation des drones dans les réseaux sans-fil

Thèse

soumise à

Sorbonne Université

pour l'obtention du grade de docteur

présentée par:

Harald Bayerlein

Soutenance de thèse effectuée le 7 septembre 2021 devant un jury composé de:

Rapporteurs		
Prof.	**Mehdi Bennis**	Université d'Oulu, Finlande
Prof.	**Andrea M. Tonello**	Université de Klagenfurt, Autriche
Examinateurs		
Prof.	**Roberto Verdone**	Université de Bologne, Italie
Prof.	**Sofie Pollin**	KU Leuven, Belgique
Prof.	**Florian Kaltenberger**	EURECOM, France
Directeur de thèse		
Prof.	**David Gesbert**	EURECOM, France

To my parents,
Waltraut and Norbert

Abstract

Autonomous unmanned aerial vehicles (UAVs), spurred by rapid innovation in drone hardware and regulatory frameworks during the last decade, are envisioned for a multitude of applications in service of the society of the future. From the perspective of next-generation wireless networks, UAVs are not only anticipated in the role of passive cellular-connected users, but also as active enablers of connectivity as part of UAV-aided networks. Use cases range from 'last mile' delivery of goods, passenger transport, infrastructure inspection, environmental monitoring and surveying to enablers of smart agriculture. Their fast and flexible deployment makes them especially useful in situations where terrestrial communication infrastructure is overwhelmed or destroyed, e.g. in natural disasters and search-and-rescue situations. In remote areas where it is not feasible or economically viable to extend permanent network infrastructure, UAVs could provide mobile Internet access to half the world's population currently without it.

The defining advantage of UAVs in all potential application scenarios is their mobility. To take full advantage of their capabilities, flexible and efficient path planning methods are necessary. This thesis focuses on exploring machine learning (ML), specifically reinforcement learning (RL), as a promising class of solutions to UAV mobility management challenges. With the recent research advances in combining RL and neural networks, deep RL is one of the few frameworks that allows us to tackle the complex task of UAV control and deployment in communication scenarios directly, given that these are generally NP-hard optimization problems and badly affected by non-convexity. Furthermore, deep RL offers the possibility to balance multiple objectives of UAV-aided networks in a straightforward way, it is very flexible in terms of the availability of prior or model information, while deep RL inference is computationally efficient.

A key limitation in path planning for small to mid-size UAV is their maximum active mission time restricted by the energy density of on-board batteries. When used as aerial base stations (BSs) providing data services to ground users, autonomous UAVs need to jointly optimize their flying time and the communication performance goals of the system. The first part of the thesis explores the use of deep Q-learning to control an aerial BS that collects data from ground users while integrating dedicated landing spots, where the UAV can land thus saving energy while continuing to serve users, in its trajectory. Deep Q-learning allows the UAV to find efficient trajectories without being given any explicit information about the environment or the mission.

While the RL paradigm offers many advantages for solving optimization problems in UAV-aided networks, there remain some practical challenges, especially in the context

of the demand for training data the UAV can learn from. Collecting training data in the real world is an expensive and time-consuming process, while in the conventional RL approach, the lengthy training process needs to be repeated if a parameter of the mission changes, e.g. the UAV's battery capacity. In this thesis, we address this issue by proposing a deep RL algorithm that is extending training to random instances of a UAV data collection mission from distributed Internet of Things (IoT) devices, where no retraining is needed if parameters of the mission change. The result is a much more complex problem compared to the conventional approach, as solutions to thousands of mission instances need to be found at the same time. This is made possible by exploiting intelligently processed map information of the dense, urban environment of the mission. We extend this setting to the cooperative multi-UAV case, where additional challenges in team coordination arise, and to large, complex and realistic city environments.

The following part of the thesis explores the connection of UAV-aided communications and robotics, two generally disjoint research communities. The inherent flexibility of the RL paradigm offers the opportunity to propose solutions that work in multiple instances of UAV path planning, such as IoT data collection and coverage path planning (CPP), a classical robotics problem. Finally, in the last part of this thesis, another approach to solving the training data demand challenge of RL algorithms based on a model-aided learning framework is examined. In this approach, the UAV learns a model of the real-world environment first, while then exploiting the learned model to generate simulated training data, reducing the demand for expensive real-world data drastically.

Abrégé

Les drones autonomes, stimulés par l'innovation rapide des technologies associées et l'évolution des cadres réglementaires au cours de la dernière décennie, sont envisagés pour une multitude d'applications au service de la société du futur. Du point de vue des réseaux sans-fil de la prochaine génération, les drones ne sont pas seulement prévus dans le rôle d'utilisateurs passifs connectés au réseau cellulaire, mais aussi comme facilitateurs actifs de la connectivité dans le cadre de réseaux assistés par drones. Les cas d'utilisation vont de la livraison de marchandises 'last mile', transport de passagers, inspection des infrastructures, surveillance de l'environnement et de l'arpentage à la mise en place d'une agriculture intelligente. Leur déploiement rapide et flexible les rend particulièrement utiles dans les situations où l'infrastructure de communication terrestre est dépassée ou détruite, par exemple lors de catastrophes naturelles et de situations de recherche et sauvetage. Dans les zones reculées où il n'est pas possible ou économiquement viable d'étendre une infrastructure de réseau permanente, les drones pourraient fournir un accès Internet mobile à la moitié de la population mondiale qui en est actuellement privée.

L'avantage déterminant des drones dans tous les scénarios d'application potentiels est leur mobilité. Pour tirer pleinement parti de leurs capacités, des méthodes de planification de trajectoire flexibles et efficaces sont une nécessité impérative. Cette thèse se concentre sur l'exploration de l'apprentissage automatique (ML), en particulier l'apprentissage par renforcement (RL), comme une classe prometteuse de solutions aux défis de la gestion de la mobilité des drones. Avec les récentes avancées de la recherche dans la combinaison de l'apprentissage par renforcement et des réseaux de neurones artificiels, l'apprentissage par renforcement profond est l'un des rares cadres qui nous permet de nous attaquer directement à la tâche complexe du contrôle et du déploiement des drones dans les scénarios de communication, étant donné qu'il s'agit généralement de problèmes d'optimisation non convexes et NP-difficile. De plus, le RL profond offre la possibilité d'équilibrer les objectifs multiples des réseaux assistés par drones de manière directe, il est très flexible en termes de disponibilité d'informations préalables ou de modèles, tandis que l'inférence RL profonde est efficace sur le plan informatique.

L'une des principales limites de la planification de trajectoire pour les drones de petite et moyenne taille est leur durée maximale de mission active, limitée par la densité énergétique des batteries intégrées. Lorsqu'ils sont utilisés comme stations de base (BS) aériennes fournissant des services de données aux utilisateurs au sol, les drones autonomes doivent optimiser conjointement leur temps de vol et les objectifs de performance de communication du système. La première partie de la thèse explore l'utilisation de deep

Q-learning pour contrôler une BS aérienne qui collecte les données des utilisateurs au sol tout en intégrant dans sa trajectoire des spots d'atterrissage dédiés, où le drone peut se poser en économisant de l'énergie tout en continuant à servir les utilisateurs. Le deep Q-learning permet au drone de trouver des trajectoires efficaces sans recevoir d'informations explicites sur l'environnement ou la mission.

Bien que le paradigme RL offre de nombreux avantages pour résoudre les problèmes d'optimisation dans les réseaux assistés par drone, il reste quelques défis pratiques, notamment dans le contexte de la demande de données d'entraînement à partir desquelles le drone peut apprendre. La collecte de données d'entraînement dans le monde réel est un processus coûteux et long, tandis que dans l'approche RL conventionnelle, le long processus d'entraînement doit être répété si un paramètre de la mission change, par exemple la capacité de la batterie du drone. Cette thèse aborde ce problème en proposant un algorithme deep RL qui étend l'entraînement à des instances aléatoires d'une collecte de données par drone à partir de dispositifs distribués de l'Internet des objets (IoT), où aucun réentraînement n'est nécessaire si les paramètres de la mission changent. Il s'agit d'un problème beaucoup plus complexe par rapport à l'approche conventionnelle, car il faut trouver des solutions à des milliers d'instances de mission en même temps. Ceci est rendu possible par l'exploitation d'informations cartographiques traitées intelligemment de l'environnement dense et urbain de la mission. La thèse étend ce cadre au cas des multi-drones coopératifs, où des défis supplémentaires en matière de coordination se présentent, et aux environnements urbains vastes, complexes et réalistes.

La dernière partie de la thèse explore la connexion entre les communications assistées par drone et la robotique, deux communautés de recherche souvent disjointes. La flexibilité inhérente au paradigme RL offre l'opportunité de proposer des solutions qui fonctionnent dans de multiples instances de recherche de chemin de drone, comme la collecte de données IoT et la recherche de chemin de couverture (CPP), un problème de robotique classique. Enfin, une autre approche pour résoudre le défi de la demande de données d'entraînement des algorithmes RL, basée sur un cadre d'apprentissage assisté par modèle, est examinée. Dans cette approche, le drone apprend d'abord un modèle de l'environnement du monde réel, tout en exploitant ensuite le modèle appris pour générer des données d'entraînement simulées, ce qui réduit considérablement la demande de données coûteuses du monde réel.

Acknowledgements

This thesis would not have been possible without the support of a large number of fantastic individuals. First and foremost, I would like to thank my advisor David Gesbert for his invaluable guidance, as well as his unconditional support during the full course of my research. I am also deeply grateful to Paul de Kerret for co-advising me during the first half of my thesis. I profited immensely from his mentorship. I thank Mehdi Bennis and Andrea Tonello for carefully reviewing this thesis and providing constructive feedback. Special thanks go to my colleagues and co-authors Rajeev Gangula and Omid Esrafilian for many great discussions. Their advice and wealth of experience were a great source of support for me. I am very grateful to Mirco Theile for the fantastic collaboration and being always able to discuss the wildest ideas with him, as well as my other collaborators at TU Munich, Marco Caccamo and Michael Koller. I also thank Akira Hirose for welcoming me into his lab at the University of Tokyo for the second time and the amazing group of people I was able to meet in Japan: thank you Eva, Jan, Jana, Jenny, Jungmin, and Yuta. I would also like to thank my first academic mentor, Michael Schöffler, and all people that have influenced me along the way of my studies and inspired me to undertake this thesis.

I feel extremely lucky that my experience at EURECOM was shaped by the great atmosphere created by all past and present members of the M3 group, my office-mates, and the amazing group of colleagues and friends I had the pleasure of getting to know during the last years. Thank you Alex, Ali, Alireza, Alison, Andreas, Anta, Antonio, Chia-Yu, Ehsan, Emanuele, Flavio, Ismail, Italo, Jonas, Judy, Konstantinos, Lorenzo, Maha, Marina, Matteo, Minhoe, Mody, Mohamad, Mounia, Naser, Omid, Placido, Pramod, Rajeev, Raphael, Riccardo, Robert, Rosa, Roya, Sagar, Shahab, Sihem, Sumit, Thomas, Xiaoguang, and everyone else who made this such a great experience. Special thanks to Guilherme who even made living through a global pandemic feel not so bad. Thanks also to all friends that have supported me from back home and from many other places around the world. Last not least, I am especially grateful to my family.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

The acronyms and abbreviations used throughout the manuscript are specified in the following. They are presented here in their singular form, and their plural forms are constructed by adding an *s*, e.g. UAV and UAVs. The meaning of an acronym is also indicated the first time that it is used. The English acronyms are also used for the abstract and summary in French.

AA	air-to-air channel
Adam	adaptive moment estimation
AG	air-to-ground channel
AI	artificial intelligence
AP	access point
BS	base station
BVLOS	beyond visual line of sight
CDMA	code-division multiple access
CNN	convolutional neural network
CPP	coverage path planning
CR	collection/coverage ratio
CRAL	collection/coverage ratio and landed
CSMA/CA	carrier-sense multiple access with collision avoidance
CTDE	centralized training and decentralized execution
DDPG	deep deterministic policy gradient
DDQN	double deep Q-network
Dec-POMDP	decentralized partially observable Markov decision process
DNN	deep neural network
DoF	degree of freedom
DH	data harvesting

DL	downlink
DP	dynamic programming
DQN	deep Q-network
DRL	deep reinforcement learning
EASA	European Union Aviation Safety Agency
FAA	Federal Aviation Administration
FL	federated learning
FLOP	floating point operation
FoV	field of view
FRAN	flying radio access network
GG	ground-to-ground channel
HAP	high-altitude platform
i.i.d.	independent and identically distributed
IoT	Internet of Things
k-NN	k nearest neighbors algorithm
LAP	low-altitude platform
LoS	line of sight
LS	landing spot
LTE	Long Term Evolution
MAC	multiple access control
MARL	multi-agent reinforcement learning
MDP	Markov decision process
MFLOP	million floating point operations
ML	machine learning
MTOM	maximum take-off mass
NFZ	no-fly zone
NLoS	non-line-of-sight
NN	neural network

NOMA	non-orthogonal multiple access
PBFT	practical byzantine fault tolerance
POMDP	partially observable Markov decision process
PSO	particle swarm optimization
QoS	quality of service
ReLU	rectified liner unit
RNN	recurrent neural network
RL	reinforcement learning
RSS	received signal strength
SC	safety controller
SGD	stochastic gradient descent
SVM	support vector machine
SINR	signal-to-interference-plus-noise ratio
SNR	signal-to-noise ratio
TD	temporal difference
TDMA	time-division multiple access
UAV	unmanned aerial vehicle
UAS	unmanned aircraft system
UE	user equipment
UL	uplink
3GPP	3rd Generation Partnership Project

Notations

The following list gives an overview of some specific notation used throughout this manuscript. Boldface uppercase letters ($\mathbf{A}$) are used for matrices or tensors, boldface lowercase letters for vectors ($\mathbf{a}$), and regular lowercase letters for scalars (a). Sets are represented by calligraphic uppercase letters ($\mathcal{A}$).

$(a_1, \ldots, a_N)$	N-tuple
$\arg\max$	Points or elements of the domain of some function at which the function values are maximized
$\lceil \ldots \rceil$	Ceiling function
$\lfloor \ldots \rceil$	Rounding to integer function
$\lfloor \ldots \rfloor$	Floor function
$\mathbb{B}$	Boolean domain $\{0, 1\}$
$\mathbb{E}_X[\ldots]$	Expectation over X
$\mathbb{R}^+$	Set of positive real numbers : $\{x \in \mathbb{R} : x > 0\}$
$\mathbf{A}^{\mathrm{H}}$	Hermitian transpose of matrix $\mathbf{A}$
$\mathbf{A}^{\mathrm{T}}$	Transpose of matrix $\mathbf{A}$
$\mathcal{N}(0,\, \sigma^2)$	Zero-mean Gaussian distribution with variance σ^2
$\times_i \mathbf{a}_i$	Joint space of vectors $\mathbf{a}_i$ over i
$\lvert\mathcal{A}\rvert$	Cardinality of set $\mathcal{A}$
$x \sim X$	Variable x follows distribution X

Chapter 1

Introduction

The technology and potential applications of unmanned aerial vehicles (UAVs), also commonly referred to as drones or unmanned aircraft systems (UAS), have undergone rapid innovation in the last decade. Crucial advances in UAV hardware, manufacturing and cost [1], in connection with newly created regulatory frameworks for the commercial use of UAVs [4], have led to the creation of a market expected to grow to $63.6 billion by 2025 [5].

Before the developments of the last decade, unmanned flight already looked back on a long history, but was mostly considered for its military applications, albeit with varying degrees of success. The first reported military use of UAVs took place in July 1849 during the Austrian siege of Venice [6]. The Austrians fitted around 100 hot-air balloons with time fuses and bombs and deployed them upwind of Venice hoping that the wind would carry them over the city. Despite thorough wind direction measurements, the balloons were scattered in all direction with a few even coming back at the Austrians themselves. No further attempts were made after this.

Fortunately, civil applications of UAVs for the society of the future have shown much more promise recently. A frequently cited scenario is the 'last mile' delivery of goods to consumers with drones. One successful commercial project of autonomous drones delivering food and groceries to customers in Iceland's capital Reykjavik was started as early as 2018, putting the Icelandic air authorities at the top of the list of countries that have allowed some form of autonomous drone flight [7]. An example in the context of keeping the world's aging infrastructure in shape is the Japanese company Hitachi, that is already commercially deploying partially autonomous UAVs which collect maintenance data from Internet of Things (IoT) sensors embedded in large structures, such as the San Juanico and Agas-Agas Bridges in the Philippines [8]. UAVs can even be used to save lives: their flexible and fast deployment make them very attractive for scenarios when stationary infrastructures (e.g. communication networks) are destroyed or unavailable, i.e. in disaster and search and rescue situations. This can be the case when a natural catastrophe like an earthquake destroys cell towers and drones act as flying base stations (BSs) to re-establish communication capabilities for first responders [9]. To fulfill their potential in support of the society of the future, it is clear that all UAVs require well-designed communication capabilities or provide communication services themselves.

1.1 UAVs in Wireless Communications

Recent years have shown a steady increase in interest for integrating UAVs into wireless communication networks, both from the commercial and academic research sphere. Fundamentally, UAVs in communication networks can be thought of in two ways and described as [1, 10]:

1. *cellular-connected UAVs* that are attached to mobile networks with a command and control link as network terminals or aerial user equipment (UE);

2. providers of communication services in *UAV-aided communication networks*, also referred to as flying radio access network (FRAN), which is also the focus of this thesis.

1.1.1 Cellular-connected UAVs

Attaching drones to cellular networks provides several benefits over other forms of connectivity, such as dedicated ground-to-UAV links. The high availability of mobile networks worldwide makes it possible to monitor autonomous UAVs or to pilot them remotely from thousands of kilometers away via cellular command and control links. At the same time it can enable UAV-to-UAV communication or the information exchange with air traffic control. Compared to traditional direct ground-to-UAV communication, cellular networks can also provide higher reliability and throughput, as well as enhanced privacy and security. Maybe most importantly, using existing cellular networks removes the requirement to set up expensive alternative infrastructure and presents a considerable advantage in cost-effectiveness [11].

Potential application scenarios are manifold, with goods delivery being the most obvious one. Large multinational companies are close to or are already deploying solutions, including Amazon's prime air delivery service using fully autonomous multirotor drones [12] or Google's Wing delivery service that was launched for a select number of Australian suburbs in 2019 [13]. Other applications such as surveillance, remote sensing and virtual reality all have similar similar network requirements for reliable, low-latency and high-speed uplink (UL) connectivity [10].

Field and experimental studies of recent years have identified the main technical challenges for cellular-connected UAVs. In [14], measurements in different altitudes showed that while more BSs are detectable at higher altitudes, the signal-to-interference-plus-noise (SINR) ratio of the best cell degrades from 150m due to downlink (DL) interference. Commercial LTE networks were shown to provide acceptable signal quality up to 120m, despite the fact that terrestrial BS antennas are tilted towards the ground [15]. Interference was also identified as the major concern 3GPP technical report TR36.777 on enhanced LTE support for UAS [16]. The report identified possibilities to detect interference at the network and/or UE side, suggests UL interference mitigation via power control, 3D beamforming, and directional UE antennas, which can be similarly applied to reduce DL interference. The potential effects of UAV mobility such as frequent handovers can be mitigated by explicitly considering the location and trajectory of the UAV. In

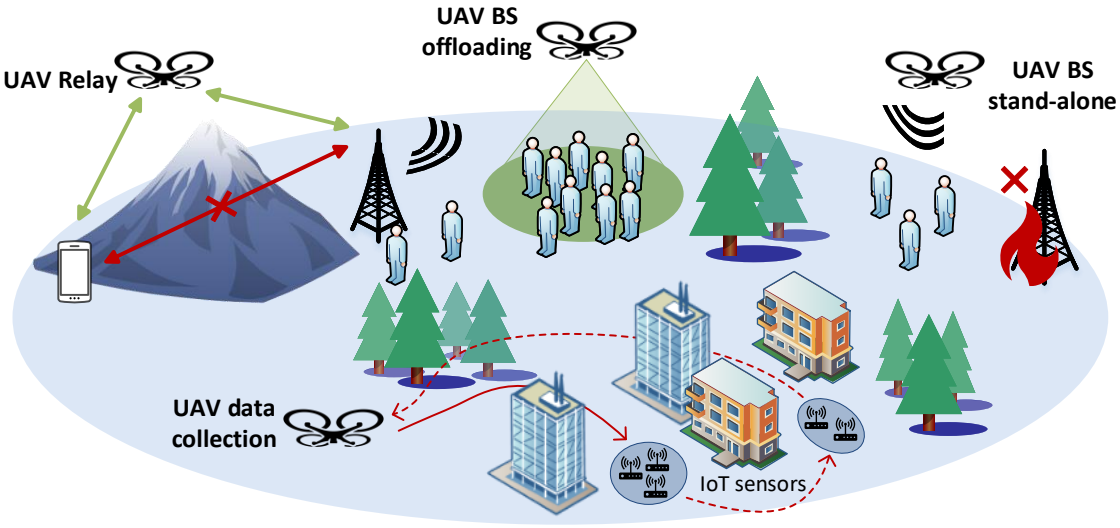


Figure 1.1 – Example applications for UAVs providing communication services and supporting stationary infrastructure.

summary, the 3GPP report concludes that small numbers of cellular-connected UAVs can be supported by existing LTE infrastructure, but that the expected exponential growth of the UAV market mandates adapted techniques. It is expected that the upcoming 3GPP TS 22.125 and TS 22.261 Release 17 concerning 5G enhancement for UAVs [17] will address some of these issues in light of 5G networks.

Concrete research challenges often revolve around UAV path planning. In [18], multiple UAVs aim to reach their respective end positions maximizing energy efficiency and minimizing both wireless latency and the interference caused on the ground network along their paths. In [19], a UAV is aiming to stay on a trajectory that guarantees reliable cellular connection while minimizing flight time. Seamless UAV connectivity is also investigated in [20,21]. Further topics include handover and resource management [22,23], protocol design [24], and UAV UE to base station association [25]. Surveys that summarize the current state of research are given by [1, 10, 26].

1.1.2 UAV-aided Communications

The advances in UAV hardware as well as the miniaturization of wireless communications equipment have enabled a multitude of potential applications for UAV-aided wireless networks, some of which are depicted in Figure 1.1. A natural use case is to attach a wireless access point (AP) as payload to a drone that then can provide additional communications capacity in areas where terrestrial networks are congested [27, 28]. Another use case of these aerial BSs are the establishment of stand-alone communication services, e.g. in areas that do not have coverage or where terrestrial infrastructure is disabled [29, 30]. UAV relays [31, 32] are likely to be able establish line-of-sight (LoS)

links to obstructed users, thanks to their high degree of mobility and altitude. Energy- and throughput-constrained Internet of Things (IoT) sensor networks can be supported by UAV data collectors that are able to describe a flight pattern that brings them into close range of devices, thereby increasing energy efficiency [33–35].

In this thesis, we focus on two use cases: an aerial BS scenario (chapter 3) and data collection from distributed IoT devices (chapters 4, 5 and 7). In the aerial BS scenario, the UAV is serving ground users at unknown locations trying to collect as much of their data in the limited mission time while granting all users equal channel access. In the IoT scenario in chapters 4 and 5, the location of the IoT devices is known, while each IoT device has a finite amount of data to be collected. In some investigated scenarios, it is possible for the UAV to collect all the data within the flying time limit, in some cases it needs to abandon a portion of the data. Chapter 7 analyzes a mixed scenario, where not all IoT device positions are known and the amount of data to be picked up at each device is unlimited.

Compared to fixed terrestrial networks, UAV-aided communication brings forward a number of unique design and research challenges [36]. Firstly, the deployment aerial BSs in 3D space provides an additional degree of freedom (DoF) compared to the typically 2D deployment of terrestrial base stations. Another major challenge is the different propagation environment [37]. The air-to-ground (AG) channel for UAVs has not been studied in too much detail previously, yet accurate propagation models are essential for developing robust communication protocols and techniques. Finally, aerial BSs and relays also add additional constraints to the maximization of communication performance or quality of service (QoS), such as limited flight time and intrinsic flight dynamics requirements. Scheduling, research allocation, and multiple access protocols need to be adapted to account for theses changes

These challenges guide the research into UAV-aided communications. Mobility can either be taken into account by UAV BS/relay placement optimization, or full end-to-end trajectory optimization. Even UAV placement is a more challenging problem compared to stationary BSs due to the additional DoF. Algorithms for UAV BS placement range from brute force [38], genetic programming [27], K-means clustering [25] to contract theory and machine learning (ML) [39]. Full trajectory optimization has been tackled, e.g. by sequential convex optimization [40], functional optimization and optimal control [41], or reinforcement learning (RL) [42]. Improving AG channel modeling requires experimental studies such as [14, 43]. Surveys and tutorials that summarize recent research are given by [1, 10, 44].

In the context of commercial 5G wireless networks, there are three scenarios that are especially relevant for UAV-aided communcations: enhanced mobile broadband (eMBB) supporting high peak data rates, massive machine-type communications (mMTC) in the context of IoT networks, and ultra-reliable and low-latency communications (URLLC) with very high reliability from a limited set of terminals [45]. While waiting for the widespread commercial implementation of delivery drones connected to 4G and 5G networks, the first considerations for what aerial communications means for 6G networks has already started [26]. Throughout the next decade, it is expected that increasing numbers of passenger air taxis, cargo UAVs, an integrated ground-air-space

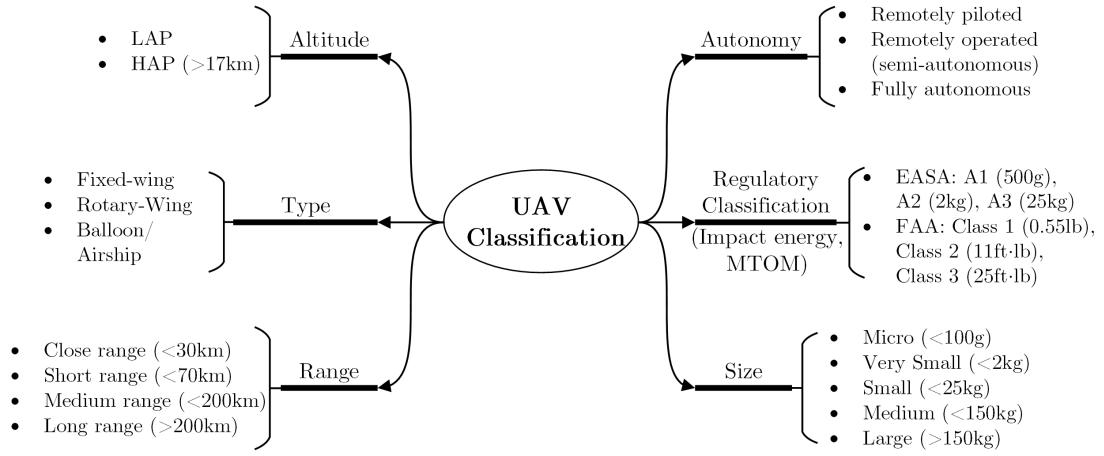


Figure 1.2 – UAV Classifications.

communication network, and cell-free architectures for UAVs, will guide the process of defining 6G requirements, making 6G a native UAV-aided communication architecture.

1.1.3 UAV Classification

While the terms UAV, UAS or drone commonly refer to any aircraft without on-board pilot, there are various ways to classify and describe UAVs according to their capabilities and intended application scenario. A possible taxonomy is depicted in Figure 1.2.

Most obviously, UAVs can be based on different flying mechanisms and classified as such. Rotary-wing drones allow for vertical take-off/landing and can hover over fixed location, e.g. the widely used quadcopter-type drones fall under this category. Fixed-wing drones can neither hover nor take-off vertically, but are typically more energy efficient and can carry a heavier payload while also typically flying at higher altitude. Balloons and airships are quasi-stationary with very long endurance and can lift heavy payloads [46].

In terms of operation altitude, UAVs are often only classified as low-altitude (LAP) or high-altitude platforms (HAP). LAPs are smaller and more flexible UAVs that can be deployed rapidly, e.g. consumer and almost all commercial drones usually fall into this category. To exceed altitudes of around 100m, most regulators require a license. HAPs on the other hand are typically deployed at altitudes above 17km for much longer missions [36]. Famous examples of HAPs are the now terminated projects Google Loon and Facebook Aquila [46]. Both projects were aiming to bring Internet connectivity to remote areas were the only current option is to use expensive and higher latency satellite connections. Google Loon aimed to launch stratospheric balloons to relay LTE signals from ground stations to cover wide areas. Aquila had a similar aim but was based on large solar-powered fixed-wing UAVs that operate at altitudes of $18 - 20$km. Both projects were eventually deemed economically unviable and cancelled in 2018 [47] and 2021 [48] respectively. The end of these means that there are currently no large commercial projects targeting HAPs for UAV-aided communication.

UAVs for the tasks in communication networks mentioned in the previous sections 1.1.1 and 1.1.2 typically fall into the categories of small size (up to 25kg) with a short or close range (up to 70km). Regulatory classifications by the European Union Aviation Safety Agency (EASA) depending on maximum take-off mass (MTOM), sort these UAVs in the open category for leisure and low risk commercial drones A1-A3 [49]. The Federal Aviation Administration (FAA) applies similar rules that partially depend on weight and partially on maximum possible kinetic impact energy [50]. The autonomy capabilities of UAVs are another important aspect and can be roughly classified from fully autonomous, i.e. UAVs that do not require any human intervention, to remotely piloted. Remotely operated or semi-autonomous UAVs might, e.g. fly independently from waypoint to waypoint, but require a human operator to decide on and define waypoints [51].

1.1.4 Regulatory Status

In addition to the advances in UAV hardware and costs, newly created regulatory frameworks that state well-defined general rules for the use of UAVs, be it for commercial or leisure activities, were an important driving force in the rapid development of the drone sector [36, 46, 52]. There are two main concerns that regulation is aiming to address. First, privacy and data protection are of high concern for UAVs, as airborne drones can easily breach personal privacy or business security through aerial surveillance, intended or not. All big regulators, including the FAA and the European Union, have put limits in place on where and when UAVs can be operated to collect any kind of data. Existing stringent data protection laws, such as General Data Protection Regulation (GDPR) in the European Union, also directly apply to UAVs and specify how legally collected data can be used, stored and processed.

The second main concern is the one of public safety, specifically the personal safety of individuals, as well as airspace safety. The wide availability of cheap end-consumer drones that are flown by mostly inexperienced pilots, poses a risk of accidents occurring through pilot or operator errors. This was addressed by eventually passing rules that define under what circumstances UAVs can be operated near of above people [49, 50]. In general, this is to be avoided in all jurisdictions and usually completely forbidden above or near large gatherings. Additionally limiting the maximum weight or kinetic impact energy of consumer drones reduces the gravity of accidents if they do occur. Collisions of drones with manned aircraft or sensitive infrastructure pose an even greater danger and can cause catastrophic accidents. This is the reason for the definition of categorical no-fly zones (NFZs), such as airports or urban city centers in UAV rule books. Virtually all jurisdictions have introduced maximum altitudes to avoid the collision of UAVs and manned aircraft in flight, typically somewhere around a maximum of 100m. The general maximum by the FAA is set at 400ft and at 120m by the EASA. National and regional regulators can be more stringent, e.g. the maximum is set as low as 50m for significant parts of France [53].

Given that the focus for UAVs in communications lies on fully autonomous drones that typically operate in the beyond visual line of sight (BVLOS) regime, aspects of regulation that are important especially for large-scale commercial use are still going

through the process of definition. Although it is not very clear as of now what future regulation of autonomous UAVs will look like, it can be assumed, based on the current rules for pilot-controlled drones, that altitude and NFZ restrictions will form part of a framework of regulation. The FAA, after having published definitive rules on flying over or near people, has now established the FAA BEYOND program [54] tackling remaining challenges of UAV integration. Specifically, these are: 1) BVLOS operations that are repeatable, scalable and economically viable with specific emphasis on infrastructure inspection, public operations and small package delivery, 2) leveraging industry operations to better analyze and quantify the societal and economic benefits of UAV operations, and 3) focusing on community engagement efforts to collect, analyze and address community concerns.

1.2 Machine Learning for UAV Communications

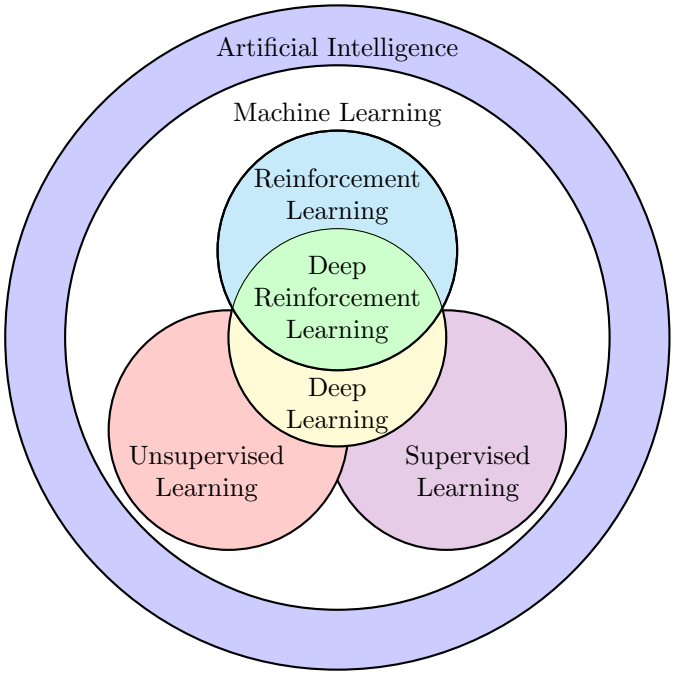


Figure 1.3 – Taxonomy of terminology in artificial intelligence and machine learning, derived from [2, 3].

The expansion of potential UAV applications and the technological progress in drone hardware also led to the emergence of new associated challenges and problems. Solutions to these challenges based on artificial intelligence (AI) and machine learning (ML) have become a highly active research area. An explosive growth of interest in both aspects of this thesis, UAV-aided communications and ML, coincided over the last years, leading to a manifold of research results that combine both. While some terminology, like AI and ML, do not necessarily have a single universally agreed definition, Figure 1.3 shows the

relationship among the most commonly used AI/ML terminology [2].

AI is the most broad term, describing the area encompassing ML. While even the 'gold standard' of AI textbooks, Russell and Norvig [55], only ventures as far as giving no less then four different definitions, the general definition of Bellman [56] defining AI as "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ...", seems at least not to raise too many objections. Machine learning, on the other hand, is an umbrella term for algorithms that improve automatically through experience and the use of data [57]. Specifically, ML algorithms are classified in the categories of supervised, unsupervised or reinforcement learning (RL). While a labeled data set of input-output pairs is available to direct the learning process in the supervised setting, unsupervised algorithms infer information from patterns in unlabeled data. In reinforcement learning, there is a form of feedback, but no explicit input-output data examples are available. RL is essentially based on exploring a problem by trial and error in settings where an agent makes sequential decisions interacting with an environment. Deep learning refers to variants of these three ML subclasses where a deep neural network (DNN) is used as part of the learning algorithm. Whether a neural network is considered 'deep' or 'shallow' is again a question of definition for which there is no consensus, but generally all neural networks with more than two hidden layers are referred to as 'deep networks'.

Supervised learning tasks often are either a form of regression, typically in the sense of prediction or forecasting, or classification. Combined regression and classification is also possible, e.g. support vector machines (SVMs) can perform both tasks. Concrete examples of supervised learning in UAV communications include trajectory prediction of UAVs relative to base station to improve communication efficiency [58], UAV-to-UAV path loss prediction using random forest and k-nearest neighbors (k-NN) algorithms [59] or the detection of UAV UEs based on LTE radio measurements [60]. The ubiquitous usage of convolutional neural networks (CNNs) in image classification has also inspired related UAV-based applications, e.g. harvest estimation in high-tech agriculture through drone imaging [61].

Algorithms that are frequently used in unsupervised learning often tackle clustering (e.g. K-means, Gaussian mixture models), dimensionality reduction (e.g. autoencoder, multi-dimensional scaling), or the generation of synthetic data (e.g. generative adverserial networks). In relation to the deployment of UAV BSs, unsupervised learning was used e.g. in the form of weighted expectation maximization (WEM) clustering to predict download traffic and deploy aerial BSs accordingly [39], or for joint mobility prediction and object profiling of UAVs to facilitate control and communication protocols [62]. In [63], clustering is used to model the 3D UAV-to-ground channel based on k-means. Unsupervised learning, specifically one-class SVM, can also be used for the detection of active eavesdropping in UAV-aided relay networks [64]. A general overview of unsupervised anomaly detection in UAVs can be found in [65].

The framework of reinforcement learning is especially suitable for challenges arising from deploying autonomous UAVs in communication networks, as the central idea is an autonomous agent taking decisions (e.g. trajectory planning) to maximize some objective (e.g. QoS for an aerial BS) in an unknown environment. Deep reinforcement learning

(DRL) can be used in many instances of UAV placement and trajectory planning, such as interference-aware multi-UAV path planning [18], data collection in the context of mobile crowdsensing [66], or maximizing communications coverage in UAV-aided networks [30]. Resource allocation is one of the UAV network applications that go beyond path planning. In [67], multi-agent reinforcement learning (MARL) is used to automatically select each UAV's communicating user, power level and subchannel without any information exchange among UAVs. An RL-based user association algorithm is developed that minimizes user hand-offs in a UAV BS network [68]. The combination of UAVs with intelligent reflecting surfaces in the context of 6G also gives rise to various challenges that can be solved by RL [26].

Another subfield of ML that has become relevant in the context of UAV communications is federated learning (FL). In deep FL, a group of decentralized edge devices collaboratively train a DNN model under orchestration of a central server [69]. Instead of exchanging all relevant data with the central server, individual devices hold on to their share of data samples, avoiding the exchange of large quantities of data and the associated privacy and data security concerns. These are issues that are also relevant for the application of ML techniques in UAV communication networks, where UAVs build a learning model collaboratively, e.g. in trajectory planning, channel modeling, or data routing and caching [70].

While ML-based approaches have shown great promise in many UAV communications applications, it is important to point out that this by no measure means that ML generally outperforms classical approaches in all scenarios and applications. In fact, ML brings its own challenges over classical approaches: sacrificing interpretability and guaranteed performance in some scenarios, while performance gains in the real world have not materialized in many areas, due to the fact that practical real-world deployment of ML-based solutions is often still difficult to achieve. However, it is also true that ML offers the chance for viable solutions to problems that are hard or impossible to tackle with classical approaches.

A wide-ranging survey on current research themes in AI for UAV-enabled wireless networks is given by [61]. Bithas et al. [71] provided an overview covering similar topics, while [52] focuses on DRL approaches.

1.3 Aims and Objectives

The most central difference of UAV-aided wireless communication networks compared to classical ones is the high degree of mobility as an additional DoF in the system. This is the defining feature that this thesis aims to explore. The advantage of UAV-aided networks essentially lies in exploiting the UAV's ability to move closer to users, avoid positions that are detrimental to communication performance, in general adapt its trajectory to serve the QoS goals of the network. Trajectory planning, an old but still complex problem, is altered from the typical aim to find the shortest or fastest path from A to B, to strike a balance between the classical path planning goals and communication network performance.

This thesis also aims to show that the problem of trajectory planning in UAV-aided

networks thereby mirrors exactly the typical formulation of an RL problem: the UAV needs to make instantaneous control decisions during its mission and receives subsequent feedback on the quality of its decisions, e.g. by achieving a certain data rate, QoS, coverage area, or a certain amount of collected data. This typically happens in a complex environment for which various parameters are subject to uncertainty, e.g. the wireless channel characteristics, structure of the environment, or exact positions of networks users. The UAV needs to explore these unknowns in the environment, while then later exploiting the knowledge it has accumulated, the classical exploration-exploitation trade-off in RL. Another important reason for the attractiveness of RL in this context is the fact that UAV control and deployment in communication scenarios are generally non-convex optimization problems [1, 30, 36, 72–74], and proven to be NP-hard in many instances [1, 73, 74], therefore difficult to solve by classical methods.

Although this work is focused on UAV communication scenarios, the inherent flexibility of RL also enables the adaptation of the described methods in other instances of UAV path planning. By making a comparison between the classical robotics problem of coverage path planning (CPP), where the drone's goal is to find a path that fully covers an area of interest, and a data collection scenario, we aim to show the interdisciplinary connection in the often disjoint research communities of robotics and wireless communications.

With this work concentrating especially on deep reinforcement learning (DRL) as a methodology to solve UAV trajectory planning problems, the well-known challenges of deploying DRL in practical real-world applications are also explored [75]. Maybe the most important challenge is thereby the need for often lengthy training phases of DRL algorithms to realize the promised performance, while training in the real-world, i.e. flying a drone, is time-consuming, complex and expensive. At the same time, state space representations of problems that are solved in the literature are often small and how to adapt the results for larger, more realistic state spaces often remains unclear. This thesis aims to explore solutions on the way to making DRL more applicable to real-world scenarios.

In many UAV communication scenarios, the expectation is that large benefits can be drawn from deploying teams of UAVs that provide communication services cooperatively. This thesis also explores the challenges that arise in cooperation and decentralization in multi-UAV systems. With reference to the previous section 1.1, the focus of this work is on UAV-aided communications while cellular-connected UAVs are not directly considered. Specifically, employing UAVs as BSs and for data collection in IoT networks are considered in detail. With reference to subsection 1.1.3, only the most common type of small rotary-wing LAP type UAVs with short to close ranges in fully autonomous scenarios are considered.

1.4 Outline and Contributions of the Thesis

Each of the following chapters explores a specific instance of UAV path planning. An overview of key modeling assumptions, available prior information, goal and problem formulations for each chapter is given in Tab. 1.1. While chapters 3, 4, 5 and 7 investigate UAV data uplink collection from IoT devices or users, chapter 6 explores the synergies of

		Chapter 3	Chapter 4	Chapter 5	Chapter 6	Chapter 7
Prior information	Channel model	✗	✗	✗	–	✗
	Environment map	✗	✓	✓	✓	✓
	Target location	✗	✓	✓	✓	partially
	Input format	Scalar	Centered map	Centered global-local maps	Centered global-local maps	Centered radio map
Multi-scenario learning		✗	✓	✓	✓	✗
Number of UAVs		Single	Single	Multiple	Single	Single
Problem formulation		MDP	MDP	Dec-POMDP	POMDP	MDP
Goal		Uplink data collection (infinite backlog)	Uplink data collection (finite backlog)	Uplink data collection (finite backlog)	Coverage path planning	Uplink data collection (infinite backlog)
TDMA scheduling		Round-robin	Max-rate	Max-rate	–	Round-robin

Table 1.1 – Comparison of key assumptions, available prior information and type of approach for the scenarios discussed in each chapter. The goal in chapter 6 is coverage path planning making the channel and scheduling category not applicable. Target locations refer to either user positions, IoT device locations, or the location of the zone of interest for coverage path planning.

UAV trajectory planning in robotics and communications. We either look at solving a single mission scenario at a time (chapters 3 and 7), or we attempt to generalize learning over multiple scenario instances at the same time (chapters 4, 5 and 6). The fact that the amount of data to be collected in chapters 3 and 7 is assumed to be unlimited (infinite data backlog) is a consequence of the unavailability of precise prior information about the users or IoT devices. Precise information is available in chapters 4 and 5, therefore the finite amount of data available at each device is known (finite data backlog). As a consequence, the TDMA scheduling is either assumed to follow the max-rate (limited data) or equal access rules (unlimited data). All instances of path planning are tackled with a DRL approach, specifically different variants of the deep Q-network (DQN) paradigm. In chapter 2, we introduce the methodology and models that are relevant for all following chapters. Specifically, this includes the grid world and UAV model, a segmented channel model for urban environments, the formulation of the general Markov decision process and an introduction to RL and deep Q-learning. The rest of this section is dedicated to giving a short description for each of the following chapters.

Chapter 3: Aerial Base Station Trajectory Planning with Landing Spots

In this chapter, we introduce the concept of using a deep Q-learning system to train a neural network to make control decisions for an autonomous UAV BS serving a group of ground users. In contrast to the following chapters, we assume absolutely no prior knowledge about the mission, the environment, or the wireless channel is available. We show that the RL framework allows the UAV agent to learn all these features while interacting with the environment. As small UAVs' mission time is strongly limited by on-board battery capacity, we introduce the concept of landing spots (LS), a small piece of real estate where the UAV can rest while continuing to serve users. Comparing the final learned trajectory to the optimal solution obtained by dynamic programming and full model knowledge, we show that the UAV agent learns to integrate LSs effectively into its trajectory, while extending its operating time and maximizing the collected data over the whole mission. The training procedure is also able to adapt to complex environmental effects, like small-scale fading and obstacle shadowing. The results of this chapter were published in:

- H. Bayerlein, P. de Kerret, and D. Gesbert, "Trajectory Optimization for Autonomous Flying Base Station via Reinforcement Learning," in Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2018 [29].

- H. Bayerlein, R. Gangula, and D. Gesbert, "Learning to Rest: A Q-learning Approach to Flying Base Station Trajectory Design with Landing Spots," in Proc. 52nd Asilomar Conference on Signals, Systems, and Computers, 2018, pp. 724–728 [76].

Chapter 4: Multi-Scenario UAV Data Harvesting in IoT Networks

While the approach in the previous chapter does not require any prior knowledge about the scenario, this comes at the price of long training time. Even more so as any change in the scenario, e.g. the position of the users, requires to rerun the full training procedure. This is also the case for the vast majority of previous work proposing DRL approaches for UAV-aided communication missions. Here, we introduce a new double deep Q-network (DDQN) method with combined experience replay for UAV trajectory planning in an IoT data harvesting scenario where the IoT devices hold a finite amount of data to be picked up by the drone. By leveraging a convolutional neural network model that exploits information about the environment from map layers, we show that we can generalize learning over multiple scenarios, i.e. the UAV agent learns to adapt instantly to significant changes in the scenario such as the number and location of IoT devices or maximum available battery capacity, without the need for lengthy retraining. Specifically, also show the large gain in learning efficiency from centering the environment maps on the UAV agent's position. These results were published in:

- H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "UAV Path Planning for Wireless Data Harvesting: A Deep Reinforcement Learning Approach," in Proc. IEEE Global Communications Conference (GLOBECOM), 2020 [77].

Chapter 5: Multi-UAV Coordination in Multi-Scenario Data Harvesting

So far, we have only analyzed single UAV scenarios which we extend in this chapter to the multi-UAV case of the IoT data harvesting scenario. This introduces new challenges in terms of coordination and multi-agent reinforcement learning (MARL) and also requires changes in the underlying Markov decision process formulation. Based on the results from the previous chapter, we also introduce a MARL approach based on centralized learning with decentralized execution (CTDE) to control a team of homogeneous UAVs on a data collection mission. We again learn a policy that can adapt without retraining to large changes in the parameters that define the data harvesting mission, such as the number of UAVs, their start positions, the number and position of IoT devices, as well as the amount of data that needs to be picked from each of them. We further adapt the approach from chapter 4 to large and complex urban environments, which necessitates an intelligent way of handling the large environment maps via dual global-local map processing. In general, the scenario in this chapter is a more complex problem with a larger state space than most of previous works have attempted to solve. We also use this chapter's approach to provide a detailed analysis of the connection between system performance and scenario parameters, as well as to compare scalar and map-based input representations and to provide some intuition about the algorithm's dependency on the channel model and the data collection performance of a stationary base station vs. the UAV-aided system. This work has led to the publication of:

- H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "Multi-UAV Path Planning for Wireless Data Harvesting with Deep Reinforcement Learning," IEEE Open Journal of the Communications Society, vol. 2, pp. 1171–1187, 2021 [35].

Chapter 6: Coverage Path Planning

Although the main focus of this thesis are UAV-aided communication scenarios, several other instances of UAV trajectory planning are closely related. In this chapter, we want to highlight the connection of IoT data harvesting to the classical robotics problem of coverage path planning (CPP). The goal for the UAV in CPP is to cover an area of interest, while adhering to obstacle avoidance and flight time constraints. The flexibility of DRL allows both problems to be solved with the same approach and neural network architecture, fundamentally only requiring a change in the reward function to reflect the change of the goal function from data collection ratio to coverage ratio. By finding a common formulation of both problems based on environment and target map layers, we show that aerial robotics and UAV communications are research areas with large unexplored synergies. We also analyze the influence of key map processing parameters of the global-local map approach on the learning performance. This work has resulted in the publications/submissions of the following articles:

- M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "UAV Coverage Path Planning under Varying Power Constraints using Deep Reinforcement Learning," in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020 [78].
- ——, "UAV Path Planning using Global and Local Map Information with Deep Reinforcement Learning," submitted to 20th International Conference on Advanced Robotics (ICAR), arXiv:2010.06917 [cs.RO], 2021 [79].

Chapter 7: Model-aided Sample-efficient UAV Trajectory Planning

In this chapter, we return to another instance of IoT data collection with limited prior information available, i.e. more than in chapter 3 but a little less than in chapters 4 and 5. Specifically, the key idea consists of using a small subset of IoT nodes as anchors (i.e. with known locations) and learning a model of the propagation environment in order to estimate the location of the other IoT nodes with unknown locations. We propose a model-accelerated DQN learning procedure that allows for fast convergence while only requiring a minimum of training data collected in the real world, as flying a drone just for training is expensive and time-consuming. By exploiting a map of the dense urban environment, learning a model of the wireless channel and introducing an IoT device localization algorithm, we achieve a reduction in real-world training data demand of at least one magnitude compared to baseline approaches with identical data collection performance. This work has been submitted to:

- O. Esrafilian, H. Bayerlein, and D. Gesbert, "Model-aided Deep Reinforcement Learning for Sample-efficient UAV Trajectory Design in IoT Networks," to be presented at IEEE Global Communications Conference (GLOBECOM), arXiv:2104.10403 [cs.IT], 2021 [80].

Chapter 2

System Model and Methodology

This chapter introduces the key concepts for the general system model pertaining to all specific instances of UAV trajectory planning in UAV-aided networks that are investigated in the following chapters. Specifically, this includes the grid world, UAV and communication channel models. Note that some level of simplification is needed when modeling the UAVs' dynamics in order to enable the implementation of the RL approach. The underlying assumptions are explicit whenever suitable.

Before moving on to a short introduction of reinforcement learning (RL), specifically Q-learning which we use in various forms in the later chapters, we describe Markov decision processes (MDPs) as the prerequisite to formulating an RL problem.

2.1 System Model

2.1.1 Grid World and UAV Model

We consider a square grid world of size $M \times M \in \mathbb{N}^2$ with cell size c and the set of all possible positions $\mathcal{M}$. A number of K communication service users is located within the limits of the grid world. The k-th user is located on ground level at $\mathbf{u}_k = [x_k, y_k, 0]^{\mathrm{T}} \in \mathbb{R}^3$ with $k \in \mathcal{K}$ where $|\mathcal{K}| = K$. Discretization of the environment is a necessary condition for the map-processing approaches presented in the following chapters, however note that all presented methods can be applied to any rectangular grid world.

As depicted in Figure 2.1, the UAV moves within the limits of the grid world $\mathcal{M}$. The UAV's mission is over after $T \in \mathbb{N}$ mission time steps, where the time horizon is discretized into equal mission time slots $t \in [0, T]$ of length δ_t seconds. The state of the UAV is thereby described through its:

- position $\mathbf{p}(t) = [x(t), y(t), z(t)]^{\mathrm{T}} \in \mathbb{R}^3$ with altitude $z(t) \in \{0, h\}$, either at ground level or in constant altitude h;
- battery energy level $b(t) \in \mathbb{R}$.

Hence, the trajectory optimization is limited to flying the UAV in a plane of fixed altitude h with only binary altitude control that allows the UAV to decide when to land. While

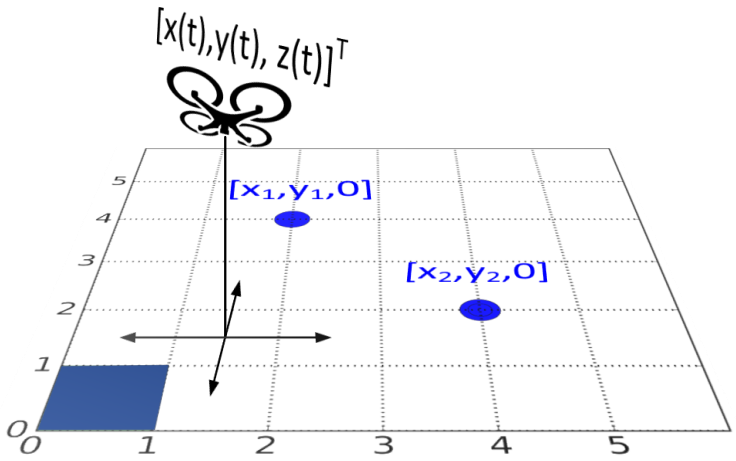


Figure 2.1 – Simplified visualization of the UAV moving through a small grid world with two ground-level users and the start/landing position marked in blue.

it is a common assumption to make in the UAV communications trajectory planning literature [1, 10, 44, 61], it seems relatively restrictive considering that in many cases the reason to use a UAV is to take advantage of altitude control as an additional degree of freedom. However, this advantage already materializes if an altitude is chosen that allows the UAV to avoid as many obstacles as possible and establish predominantly LoS links with the target communication devices, not necessarily requiring active altitude control during the mission.

As explained in section 1.1.4, it is also not very clear as of now what future regulation of autonomous UAVs will look like, but it can be assumed, that altitude restrictions will form part of a framework of regulation. Current regulation in France on flying small quadcopter-type drones sets the maximum altitude at 50m for significant parts of France [53]. For the dense urban scenarios we consider later, this would mean that even with full altitude control, UAVs would not be able to fly over the significant number of high-rise buildings, therefore not being able to take full advantage of the additional degree of freedom in any case.

Furthermore, UAVs are typically severely restricted by their on-board battery budget and rely on energy-efficient flight to maximize their active mission time. Climbing flight consumes more energy than horizontal flight [81], impacting the mission duration and therefore communication performance. For the mentioned reasons, the focus of this work is on designing 2D trajectories.

We consider different power consumption models and discrete action spaces available to the UAV in different scenarios, which is why those will be introduced in the specific chapters later on.

2.1.2 Communication Channel Model

In UAV-aided communications, essentially three types of channels exist: the air-to-ground (AG) channel, the air-to-air (AA) channel, and the ground-to-ground (GG) channel. While the GG channel is well investigated, the AA or UAV-to-UAV channel can usually be modelled by simple free space loss due to the unobstructed propagation environment [82]. The AG or UAV-to-ground channel on the other hand, exhibits distinct characteristics in comparison to terrestrial communication channels [37].

The inherent advantage of the AG channel over the GG channel is the increased likelihood of establishing LoS connections with ground users due to the UAV altitude. Even under NLoS conditions it is likely that the AG channel incurs smaller diffraction and shadowing losses than near-ground terrestrial links. The high degree of UAV mobility, on the other hand, leads to faster changes and fluctuations compared to GG channels. As rotary-wing type UAVs have the possibility to hover however, it is also possible that adverse propagation conditions, e.g. caused by local fading, last several seconds [45].

The following channel model will be used for all following chapters of this thesis, except chapter 3, where we used a simplified but similar model. Due to the focus of this work on environments with many obstructions, i.e. urban environments, we follow an LoS/NLoS segmented UAV-to-ground channel model. Consequently, the communication links between UAVs and the K users are modeled as LoS/NLoS point-to-point channels with log-distance path loss and shadow fading. The information rate at time t for the k-th user is given by

$$R_k(t) = \log_2\left(1 + \mathrm{SNR}_k(t)\right). \tag{2.1}$$

The SNR with transmit power P_k, white Gaussian noise power at the receiver σ^2, UAV-user distance d_k, path loss exponent α_e and $\eta_e \sim \mathcal{N}(0, \sigma_e^2)$ modeled as a Gaussian random variable, is defined as

$$\mathrm{SNR}_k(t) = \frac{P_k}{\sigma^2} \cdot d_k(t)^{-\alpha_e} \cdot 10^{\eta_e/10}. \tag{2.2}$$

The existence of propagation obstacles hindering free propagation causes a strong dependence of the propagation parameters on the $e \in \{\text{LoS, NLoS}\}$ condition and note that (2.2) is the SNR averaged over small scale fading. The UAV trajectory optimization time scale is usually much longer than the fast fading coherence time, hence fast fading can be averaged out in approximation. It is also important to note that the RL-based trajectory planning algorithms we develop in later chapter are model-free and therefore do not rely on any specific channel model. While a more accurate and complex model could be directly used with RL approach, the most important features for UAV communication mission in urban environments, the dependence of the SNR on d_k and the $e \in \{\text{LoS, NLoS}\}$ condition, are already captured in (2.2). A more detailed analysis of the channel model dependency and the description of multiple access control (MAC) protocols follows in the later chapters.

2.2 Markov Decision Process

Markov decision processes (MDPs) are a mathematically idealized form of the reinforcement learning problem and as such, form the basis of RL [83, 84]. This section introduces the key concepts of finite-horizon, discrete-time, stochastic MDPs. We then extend the basic MDP formulation in the following chapters as required for the specific scenario. While infinite-horizon MDP formulations are also possible, UAV-aided communication scenarios typically have a well-defined end-point, i.e. when the UAV's battery runs out and it must return to a charging station. Stochastic MDPs are an extension of deterministic state transitions: in a stochastic MDP, the next state is not deterministically given by the current state and action. Instead, the next state is a random variable, and the current state and action give the probability density of this random variable [85]. In terms of notation, we follow the example of Puterman [86], one of the standards in the MDP literature. As such and in line with standard MDP convention, the time index t is written in subscript in the following, but will be omitted in later chapters.

MDPs can be used to formalize a discrete time planning task of a single agent in a stochastically changing environment, on the condition that the agent can observe the state of the environment. It is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R)$ with state-space $\mathcal{S}$, action space $\mathcal{A}$, state transition function P and reward function R:

- The state space $\mathcal{S}$ defines the current condition of the environment, i.e. a finite set of states, where a state is a unique characterization of all that is important for the problem that is modeled. The state typically includes features like the current position and battery content of the drone.

- A finite set of actions is defined by the action space $\mathcal{A}$. Actions can be used to control the system state, e.g. the movement directions that are available to the UAV. In some systems, not all actions can be applied in every state, e.g. the safety controller of a drone can block unsafe actions to prevent accidents.

- The probabilistic state transition function is given by $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, i.e. the probability of ending up in state s_{t+1} after executing action a_t in state s_t is denoted $P(s_{t+1}|a_t, s_t)$. P is therefore equivalent to a probability distribution over possible next states.

- The reward function, that defines the UAV mission goals and is a measure of its performance, maps state-action pairs to a real-valued reward, i.e. $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

The Markov property is fulfilled if the effect of an action only depends on the current state and not on the history of past actions and visited states, i.e.

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ...) = P(s_{t+1}|s_t, a_t), \tag{2.3}$$

implying that the current state gives enough information to make an optimal decision for the next action.

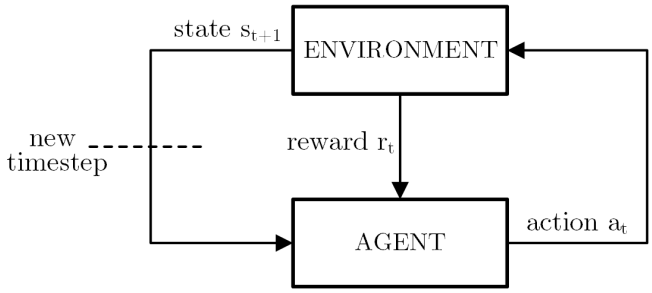


Figure 2.2 – The RL interaction cycle between agent and environment.

2.3 Reinforcement Learning

2.3.1 Agent-Environment Interaction Cycle

Reinforcement learning (RL) in general proceeds in a cycle of interaction between an agent and the environment as depicted in Figure 2.2. The environment enables the agent to learn and optimize a behavior, i.e. the agent observes state $s_t \in \mathcal{S}$ and performs an action $a_t \in \mathcal{A}$ at time t. The environment subsequently assigns a reward $r(s_t, a_t) \in \mathbb{R}$ to the agent. Then, the cycle restarts with the propagation of the agents to the next state s_{t+1}. The agent's goal is to learn a behavior rule, referred to as a policy that maximizes its reward. Different RL algorithms exist that can be used to learn a policy, such as Q-learning, explained in detail in the following section.

Another important question to consider in this context is the agent-environment boundary: what element of the system is actually considered part of the agent or the environment respectively? Typically, it is not the same as the physical boundary between e.g. the drone and the surrounding environment, but instead the mechanical and sensor hardware of the drone should already be considered part of the environment. The general rule is that anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus forms part of its environment [83]. This does not necessarily mean that everything concerning the environment is unknown to the agent, e.g. having some idea about how the reward is computed is essential, but as it is beyond the control of the agent, the reward is considered part of the environment. This also does not exclude the possibility that the agent-environment boundary changes for different tasks, e.g. if an agent has to make high- and low-level decisions, the boundary can include different parts of the system for different purposes.

2.3.2 Q-learning

Q-learning is an off-policy RL method proposed first by Watkins in 1989 [87] and developed further in 1992 [88]. It is classified as off-policy because the decisions are made by a policy that is different from the one that is being learned. Q-learning specifically allows an agent to learn to act optimally in an environment that can be represented by an MDP. In such an environment, Q-learning is proven to find a reward-optimal mapping between states and actions given that all actions in all states are repeatedly sampled [88].

This guarantee does unfortunately not hold if function approximation, such as a neural network (NN) is used.

In order to learn to act optimally, the agent learns a behavior rule or a policy that maximizes its reward. A probabilistic policy $\pi(a|s)$ is a distribution over actions given the state such that $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. In the deterministic case, it reduces to $\pi(s)$ such that $\pi : \mathcal{S} \to \mathcal{A}$.

To judge 'how good' it is for the agent to be in a certain state, the value of a state s under policy π is given as

$$V^{\pi} = \mathbb{E}_{\pi}\left[G_t \mid s_t = s\right] \tag{2.4}$$

representing an expectation of the the discounted cumulative return G_t from the current state s_t up to a terminal state at time T, which is given by

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} r(s_k, a_k) \tag{2.5}$$

with $\gamma \in [0, 1]$ being the discount factor, balancing the importance of immediate and future rewards. Similarly, Q-learning is based on iteratively improving the state-action value function or Q-function to guide and evaluate the process of learning a policy π. It is given as

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[G_t \mid s_t = s, a_t = a\right]. \tag{2.6}$$

Value functions fundamentally satisfy a recursive property as given by the *Bellman Equation* [89]

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \mid s_t = t\right] \\ &= E_{\pi}\left[r_t + \gamma V^{\pi}\left(s_{t+1}\right) \mid s_t = s\right] \\ &= \sum_{s'} P\left(s, \pi(s), s'\right)\left(R\left(s, a, s'\right) + \gamma V^{\pi}\left(s'\right)\right). \end{aligned} \tag{2.7}$$

It denotes that the expected value of state s is defined in terms of the immediate reward and values of possible next states weighted by their transition probabilities, and additionally a discount factor. V^{π} is the unique solution for this set of equations. Note that multiple policies can have the same value function, but for a given policy π, V^{π} is unique [84].

The optimal Q-function for a given MDP is given by

$$Q^{*}(s, a) = \sum_{s'} P\left(s, a, s'\right)\left(R\left(s, a, s'\right) + \gamma \max_{a'} Q^{*}\left(s', a'\right)\right) \tag{2.8}$$

with the optimal policy directly following as

$$\pi^{*}(s) = \arg\max_{a} Q^{*}(s, a). \tag{2.9}$$

This clarifies why the Q-function is useful: no forward reasoning step is needed to decide on an optimal action and P and R can be unknown. If the Q-function is known, the best action is always the one that maximizes the Q-function.

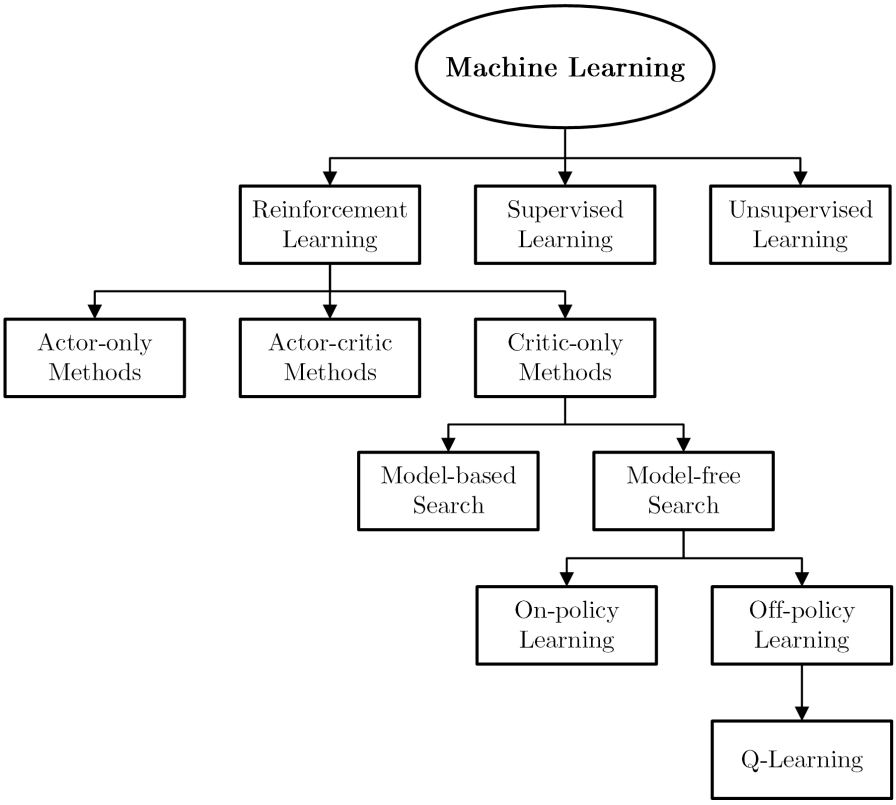


Figure 2.3 – Classification of Q-learning in the context of a taxonomy of RL methods.

In other forms of RL algorithms, the Q-function does not always play a role. Referring to the taxonomy depicted in Figure 2.3, Q-learning is a critic-only RL method relying on an indirect way to find a behavior policy by iteratively improving on the approximation of the state-action value function. In contrast, actor-only learning methods iterate directly over the policy space. Actor-critic methods are a combination of both paradigms. As the goal in Q-learning is not to directly learn a model of the environment, i.e. the MDP, it is a model-free method. Model-based RL explicitly aims to learn the MDP and then uses a different kind of planning algorithm on the MDP to solve the problem.

As Q-learning does not follow the learned policy during exploration of the state space, it is categorized as off-policy, which means that the learned state-action value function Q^{π} directly approximates Q^{π^*}, independently of the policy being followed. The policy still has the effect of controlling which state-action pairs are visited and updated during the process of exploration. Learning off-policy is a necessary condition for training the neural network as explained in the following section.

Furthermore, Q-learning can be seen as an instance of interactive learning as the agent can directly influence its observations and thereby the distribution of training data. Additionally, Q-learning is an instance of an *online learning* as feedback is given through the reward signal during the learning process, as opposed to methods that only produce a policy after a defined minimum number of samples is available [90].

2.3.3 Deep Q-learning

For ease of exposition, s_t and a_t are abbreviated to s and a, while s_{t+1} and a_{t+1} are abbreviated to s' and a' in the following. As demonstrated in [29], representing the Q-function (2.6) as a table of values is not efficient in the large state and action spaces of UAV trajectory planning. Instead, a neural network parameterizing the Q-function with the parameter vector θ can be trained to minimize the expected temporal difference (TD) error given by

$$L(\theta) = \mathbb{E}_\pi[(Q_\theta(s,a) - Y(s,a,s'))^2] \tag{2.10}$$

with target value

$$Y(s,a,s') = r(s,a) + \gamma \max_{a'} Q_\theta(s',a'). \tag{2.11}$$

While a neural network is significantly more data efficient compared to a Q-table due to its ability to generalize, the *deadly triad* [83] of function approximation, bootstrapping and off-policy training can make its training unstable and cause divergence. These problems become more serious with larger networks, which are called deep Q-networks (DQNs).

Through the work of Mnih *et al.* [91] on the application of techniques such as experience replay, it became possible to train large DQNs stably. Experience replay is a technique to reduce correlations in the sequence of training data. New experiences made by the agent, represented by quadruples of (s,a,r,s'), are stored in the replay memory $\mathcal{D}$. During training, a minibatch of size m is sampled uniformly from $\mathcal{D}$ and used to compute the loss.

In addition to experience replay, Mnih *et al.* used a separate target network for the estimation of the next maximum Q-value, giving the loss as

$$L^{\mathrm{DQN}}(\theta) = \mathbb{E}_{s,a,s'\sim\mathcal{D}}[(Q_\theta(s,a) - Y^{\mathrm{DQN}}(s,a,s'))^2] \tag{2.12}$$

with target value

$$Y^{\mathrm{DQN}}(s,a,s') = r(s,a) + \gamma \max_{a'} Q_{\bar{\theta}}\left(s',a'\right). \tag{2.13}$$

$\bar{\theta}$ represents the parameters of the target network. The parameters of the target network $\bar{\theta}$ can either be updated as a periodic hard copy of θ or with a soft update

$$\bar{\theta} \leftarrow (1-\tau)\bar{\theta} + \tau\theta \tag{2.14}$$

after each update of θ. $\tau \in [0,1]$ is the update factor determining the adaptation pace. Figure 2.4 summarizes the DQN training process.

2.3.4 Exploration-Exploitation Dilemma

At the heart of any RL algorithm like Q-learning lies a fundamental dilemma: the exploration-exploitation trade-off. It is a specific characteristic of RL and does not appear in other forms of learning. Fundamentally, the RL agent must interact with environment to learn, i.e. learning by trial-and-error, that means exploring the environment and perceiving the consequences in the form of the reward signal. However, the agent also has to exploit its accumulated knowledge in order to increase the reward it obtains.

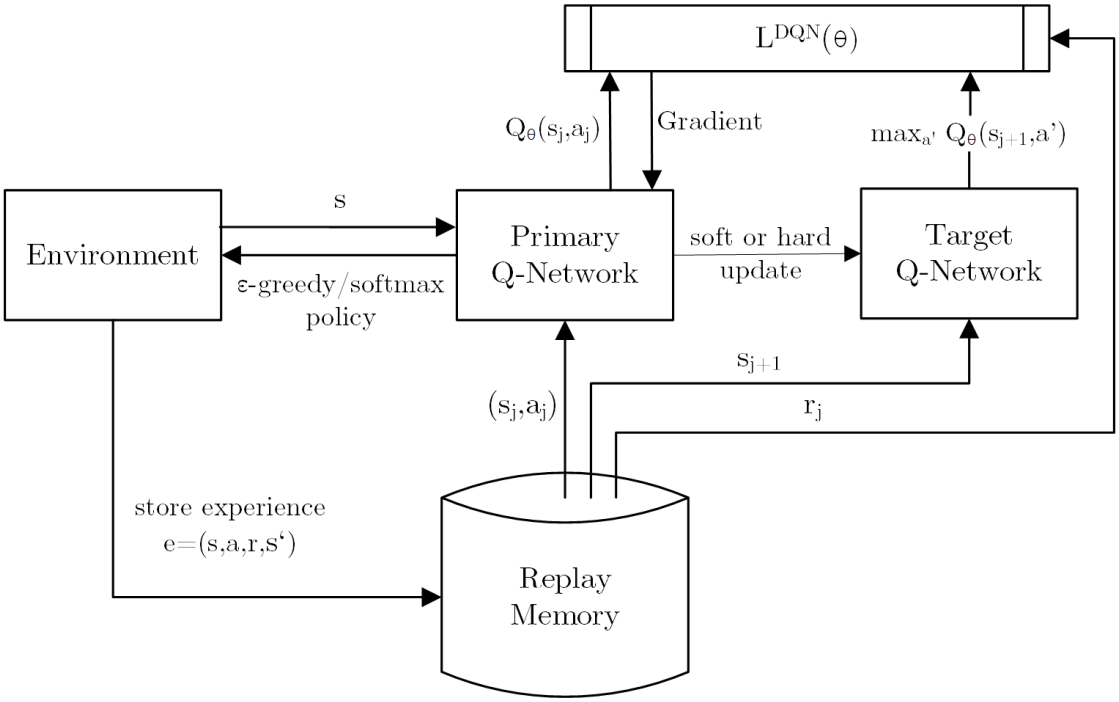


Figure 2.4 – The training process of DQN with experience replay and target network.

Additionally, in a stochastic environment, every action must be selected multiple times to gain any kind of meaningful estimate of the expected reward. How long should the agent keep exploring before exploiting its knowledge? This is the exploration-exploitation dilemma, which has been studied intensively in mathematics, yet remains unresolved [83].

In RL, the exploration-exploitation trade-off is balanced through the use of exploration strategies. These strategies introduce a certain amount of randomness in the action selecting policy, recalling (2.9) that states that the optimal policy for perfect knowledge of the Q-function is the greedy policy. The randomness fulfills the function of exploring the state-action space and is reduced gradually over the learning time of the agent to allow for a gradual increase in exploitation.

One commonly used exploration policy is the ϵ-greedy policy, where with probability ϵ, an action is randomly selected from the action space $\mathcal{A}$ and otherwise the action that maximizes the Q-function in the current state. The ϵ-greedy policy is given by

$$\pi(a|s) = \begin{cases} \text{randomly select from } \mathcal{A}, & \text{with probability } \epsilon \\ \arg\max_{a\in\mathcal{A}} Q_\theta(s,a), & \text{otherwise.} \end{cases} \tag{2.15}$$

The ϵ-greedy policy is simple, but is as likely to choose the worst-appearing action as it is to choose the next-to-best action. Another commonly used strategy, the soft-max policy, avoids this problem by varying the action probabilities as a graded function of

estimated value. The soft-max policy given by

$$\pi(a|s) = \frac{\mathrm{e}^{Q_\theta(s,a)/\beta}}{\sum_{\forall a_i \in \mathcal{A}} \mathrm{e}^{Q_\theta(s,a_i)/\beta}}. \tag{2.16}$$

This is also referred to as Boltzmann exploration as the agent draws from a Boltzmann distribution. The temperature parameter $\beta \in \mathbb{R}$ scales the balance of exploration versus exploitation.

Chapter 3

Aerial Base Station Trajectory Planning with Landing Spots

3.1 Introduction

One promising idea in UAV-aided networks is to leverage the versatility of drones for the mobile communication infrastructure itself. Deploying mobile base stations (BSs) mounted on UAVs could provide network operators with the capability to react fast and efficiently to sudden demand increases in localized areas, e.g. caused by crowded events, as well as immediately re-establish destroyed networks in disaster and search-and-rescue scenarios. Alternative carrier systems such as fixed-wing aircraft or balloons could also be used to establish network coverage and Internet connectivity in areas without fixed infrastructure.

No matter the scale of the established network, the Quality of Service (QoS) afforded to the network's users is strongly dependent on the location of the aerial BS. Previous works either addressed the placement problem of finding a drone position that maximizes the system's QoS goals, e.g. in [92, 93], or the trajectory planning problem where the drone's flying path from start to end is optimized with respect to the QoS goals, e.g. in [29, 40, 94, 95]. When only addressing the placement problem, the performance while flying to and from the designated position is not being optimized. In [94], considering the whole trajectory allows the authors to jointly optimize scheduling and user association, whereas the authors of [40] and [95] consider the power consumption of the UAV and attached BS in addition to the QoS.

As this work focuses on small and versatile multirotor type drones, of which the quadcopter is the most commonly used example, power consumption and battery energy density restrictions are central constraints for UAV BS mission planning. A typical quadcopter fitted with a small base station as used in experiments at EURECOM [96] can only sustain a mission duration of around 15 minutes. Even when turning a quadcopter into a "flying battery", maximum mission durations can usually not exceed two hours [97]. As power consumption for flying usually exceeds power consumption of the carried BS by far, the concept of landing spots (LSs) was introduced in [95] to extend mission duration.

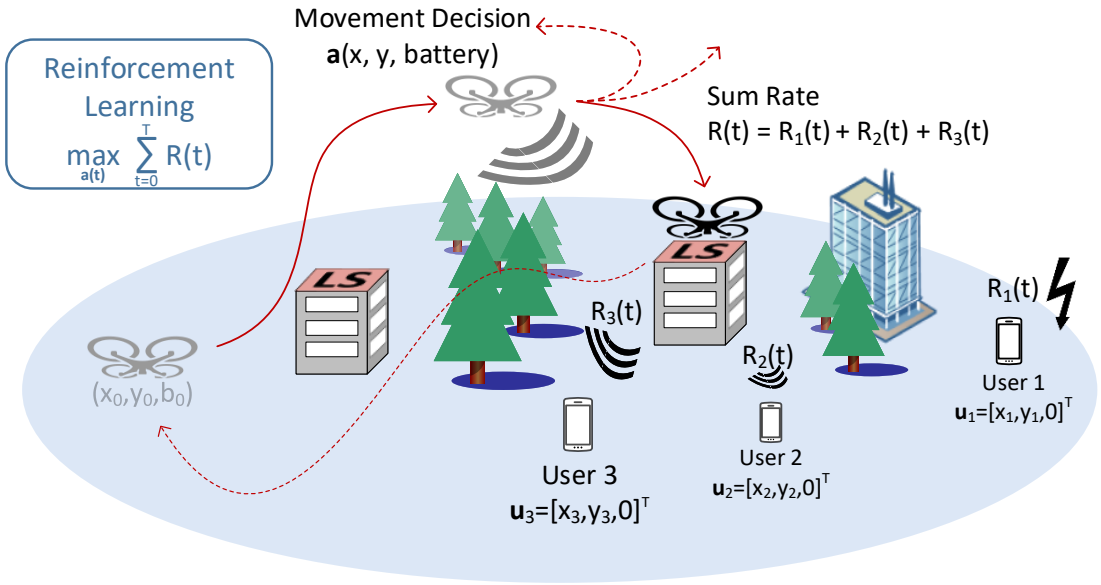


Figure 3.1 – Aerial BS movement decisions are made based on the drone's current state, i.e. position and battery content. The UAV has absolutely no prior knowledge of the environment, i.e. no knowledge of the existence or location of LSs, of user positions, of the channel model, or of the final UAV landing position. While LSs offer the possibility to conserve energy, the UAV BS might have to sacrifice some QoS for some users.

A landing spot is a small piece of real estate, e.g. a roof, where a UAV BS can land thus saving energy while continuing to serve users. Often, the utilization of the LS goes along with the sacrifice of some instantaneous QoS for some users, but helps to extend the overall mission duration.

As introduced in sections 2.3.2 and 2.3.3, Q-learning is now a widely used RL technique in UAV communications [52,61,98]. Since the seminal paper by Mnih et al. [91], especially the deep Q-network (DQN) paradigm that allows for stable training of larger networks has found many applications in UAV trajectory planning. Two of the earliest examples are [99] and [29]. In [99], a UAV is tasked with collecting data in the context of smart cities with the help of charging stations. The UAV agent makes movement decisions under the guidance of a DQN. In [29], the concept of UAV BS trajectory optimization with DQN is introduced, while making a comparison that showcases the advantages of DQN over table-based Q-learning in the large state-action spaces of UAV trajectory optimization, however without any consideration of power consumption and LSs.

In this chapter, we consider the UAV acting as a mobile BS serving a group of ground users maximizing the sum of the information rate over the whole flying time with a limited amount of energy in the drone's battery at the start. Movement decisions are therefore made based on the UAV's current position and battery content, as well as the expectation of the total sum rate that can be achieved until the battery has run out. To save energy during the mission, the UAV is allowed to land in designated LSs as depicted

in Fig. 3.1.

It is important to note that, also in contrast to the scenarios described in later chapters of this thesis, the UAV here has absolutely no prior knowledge of the environment or the mission scenario. That means no knowledge of the existence or location of LSs, of the number and location of users, of obstacles, of the channel model, or of the final UAV landing position. With reference to the exploration-exploitation dilemma (see section 2.3.4), this means the UAV must discover all environmental features by trial-and-error. We also compare the data collection performance of the DQN system with the optimal trajectory that can be computed using dynamic programming (DP) based on an approach along the lines of [95]. In contrast, the DP approach of course requires complete prior knowledge of the environment and the model.

3.2 Optimization Problem

In reference to section 2.1.1, we consider the UAV BS inside a square grid world $\mathcal{M} = [0, M] \times [0, M]$ with cell size c serving K ground users located at positions $\mathbf{u}_k = [x_k, y_k, 0]^{\mathrm{T}} \in \mathcal{M}$ with $k \in \{1, ..., K\}$. The LSs and their locations are given by the set

$$\mathcal{L} = \left\{ \left[x_i^l, y_i^l\right],\ i = 1, \ldots, L,\ : \left[x_i^l, y_i^l\right] \in \mathcal{M} \right\}.$$

3.2.1 UAV Model

The UAV starts its mission from an initial position $\mathbf{p}_0 = [x_0, y_0, 0]^{\mathrm{T}}$ and is assumed to travel at constant altitude h with a maximum velocity V, i.e. $v(t) \leq V$. The mission is over when the drone's battery is empty defined as time T, by which the UAV is supposed to be in the final position $\mathbf{p}_f = [x_f, y_f, 0]^{\mathrm{T}}$. During the mission, $t \in [0, T]$, the drone's position is given by $\mathbf{p}(t) = [x(t), y(t), z(t)]^{\mathrm{T}}$ with altitude $z(t) \in \{0, h\}$. As we basically look at a 2D problem, the UAV's position is defined by the functions $x(t)$ and $y(t)$ given as

$$x : \begin{pmatrix} [0, T] \to \mathbb{R} \\ t \to x(t) \end{pmatrix} \qquad y : \begin{pmatrix} [0, T] \to \mathbb{R} \\ t \to y(t) \end{pmatrix} \tag{3.1}$$

subject to

$$x(0) = x_0, \qquad y(0) = y_0 \tag{3.1a}$$

$$x(T) = x_f, \qquad y(T) = y_f \tag{3.1b}$$

As The UAV battery's energy content is denoted by

$$b(t) \geq 0, \qquad \forall t \in [0, T] \tag{3.2}$$

during mission time with a full charge when the mission begins, i.e. $b(0) = b_{max}$. Power consumption of the autonomous UAV BS is modeled by a flying and mobility component $p_f(t)$, as well as a communication and computation component $p_c(t)$ which is assumed to be constant during the mission, i.e.

$$p_c(t) = p_c, \qquad \forall t \in [0, T]. \tag{3.3}$$

Energy usage for flying is constant as well, except when the drone has landed in a designated LS, i.e.

$$p_f(t) = \begin{cases} 0, & \forall t \ : [x(t), y(t)] \in \mathcal{L}, \\ p_f, & \text{otherwise.} \end{cases} \tag{3.4}$$

It follows that the UAV's battery content evolves according to

$$b(t+1) = b(t) - p_f(t) - p_c \tag{3.5}$$

over the whole mission time $t \in [0, T]$ of the aerial BS

3.2.2 Communication Channel Model and Maximization Problem

The communication links between UAV BS and the group of K users are modeled as orthogonal point-to-point channels. We deviate here from the model presented in section 2.1.2 by considering random small-scale Rayleigh fading and a constant attenuation factor under NLoS condition, as the goal of this work is also to allow for direct comparison with the dynamic programming approach from [95] and the previous work in [29]. The work in subsequent chapters will however use the more realistic model as presented in section 2.1.2. Consequently, the information rate for the k-th user, $k \in \{1, ..., K\}$ located at static position $\mathbf{u}_k \in \mathcal{M}$ at ground level is given by

$$R_k(t) = \log_2 \left(1 + \mathrm{SNR}_k(t)\right), \tag{3.6}$$

where the signal-to-noise ratio (SNR) with transmit power P_k, UAV-user distance $d_k(t)$ and path loss exponent $\alpha = 2$, is defined as

$$\mathrm{SNR}_k(t) = \frac{P_k}{\sigma^2} \cdot d_k(t)^{-\alpha} \cdot X_{Rayleigh} \cdot \beta_{shadow}. \tag{3.7}$$

Rayleigh fading was modeled as a random variable $X_{Rayleigh} \sim \exp(1)$. Attenuation through obstacle obstruction is a discrete factor $\beta_{shadow} = 0.01$ under NLoS conditions, and $\beta_{shadow} = 1$ everywhere else.

Using the described model, the maximization problem can be formulated as

$$\max_{x(t), y(t)} \sum_{t=0}^{T} \sum_{k=1}^{K} R_k(t) \tag{3.8}$$

subject to aforementioned constraints (3.1a), (3.1b), (3.2) and (3.5).

3.3 Neural Network Training and Algorithm

3.3.1 Markov Decision Process

With reference to section 2.2, we transform the maximization problem (3.8) into an MDP that allows us to solve it with an RL approach. The MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R)$ with state-space $\mathcal{S}$. In this scenario, we assume a deterministic state transition function $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$.

State Space

The state space $\mathcal{S}$ is defined solely through the UAV agent's own state. It is given as

$$\mathcal{S} = \underbrace{\mathbb{R}^3}_{\substack{\text{UAV} \\ \text{position}}} \times \underbrace{\mathbb{R}}_{\substack{\text{Battery} \\ \text{content}}} \times \underbrace{\mathbb{B}}_{\substack{\text{LS} \\ \text{detection}}} \tag{3.9}$$

in which the elements $s(t) \in \mathcal{S}$ are

$$s(t) = (\{\mathbf{p}(t)\}, \{b(t)\}, \{bool_{LS}(t)\}), \tag{3.10}$$

where $bool_{LS}(t)$ is a boolean variable that indicates whether the UAV is currently in a LS position. In practice, only the 2D position of the drone is relevant for movement decisions during the mission. Therefore, $\mathbf{p}(t)$ is now defined as the UAV's 2D position projected on the ground.

Action Space

The action space of the aerial BS agent is defined as

$$\mathcal{A} = \left\{ \underbrace{\begin{bmatrix}0\\0\end{bmatrix}}_{\text{hover/land}}, \underbrace{\begin{bmatrix}0\\c\end{bmatrix}}_{\text{north}}, \underbrace{\begin{bmatrix}c\\c\end{bmatrix}}_{\text{northeast}}, \underbrace{\begin{bmatrix}0\\c\end{bmatrix}}_{\text{east}}, \underbrace{\begin{bmatrix}-c\\c\end{bmatrix}}_{\text{southeast}}, \underbrace{\begin{bmatrix}-c\\0\end{bmatrix}}_{\text{south}}, \underbrace{\begin{bmatrix}-c\\-c\end{bmatrix}}_{\text{southwest}}, \underbrace{\begin{bmatrix}0\\-c\end{bmatrix}}_{\text{west}}, \underbrace{\begin{bmatrix}c\\-c\end{bmatrix}}_{\text{northwest}}, \right\}. \tag{3.11}$$

If the agent decides to stand still, this is implicitly defined as landing if the UAV is currently in a LS location, or as hovering if not in a LS location.

Reward Function

The reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is comprised of the following elements:

$$r(t) = \sum_{k=1}^{K} R_k(t) + \rho(t) + \kappa(t) + \xi(t), \tag{3.12}$$

where the main element is the sum information rate of all users in the current time slot. The other elements are defined as follows:

$$\rho(t) = \begin{cases} -10, & \text{if } x(t) > M \vee y(t) > M \\ 0, & \text{otherwise} \end{cases} \tag{3.13a}$$

$$\kappa(t) = \begin{cases} -\sum_{k=1}^{K} R_k(t) - 10, & \text{if } \|(x(t), y(t)) - (x_f, y_f)\|_1 = V(T-t) \\ 0, & \text{otherwise} \end{cases} \tag{3.13b}$$

$$\xi(t) = \begin{cases} 10 & \text{if } (x(t), y(t)) \in \mathcal{L} \text{ and first visit} \\ 0, & \text{otherwise,} \end{cases} \tag{3.13c}$$

where $\rho(t)$ represents a punishment for leaving the assigned mission area, i.e. stepping out of the 11 by 11 grid world, $\kappa(t)$ is activated as a punishment if the UAV's safety

controller detects that the final position is not reachable anymore within the given flight time limit, and $\xi(t)$ is a special one-off reward that is given to the agent for discovering a new LS. As the numerical value of these punishments and rewards can be chosen freely by the environment designer, it is reasonable to fix them to values in the same range as the expected reward from the sum rate, the reward signal's main component. If tested in more diverse scenarios then it is the case here, the numerical rewards and punishments should be normalized.

3.3.2 Neural Network Model

In traditional Q-learning, the Q-function is usually represented by a multidimensional table that contains one Q-value for each state-action pair. It becomes immediately evident that this is not practical in large state and action spaces as the table size would grow exponentially when adding space dimensions. A way out of this dilemma is the use of a DQN (see section 2.3.3 which approximates the optimal Q-function $Q^*(s, a)$ by a neural network $Q_\theta(s, a)$ with network parameters θ. The main advantages of this approach include the DQN's ability to generalize from few data samples to the whole state space and the higher training data efficiency. A more detailed comparison of table- and NN-based Q-learning for UAV BS trajectory optimization can be found in [29].

Fig. 3.2 shows the DQN architecture used in this work. One state space sample containing drone position, current battery content and landing spot availability forms the input. Two fully connected hidden layers with $n = 100$ units each are followed by the output layer with outputs equal to the cardinality of the drone's action space. All neurons are rectified linear units (ReLU).

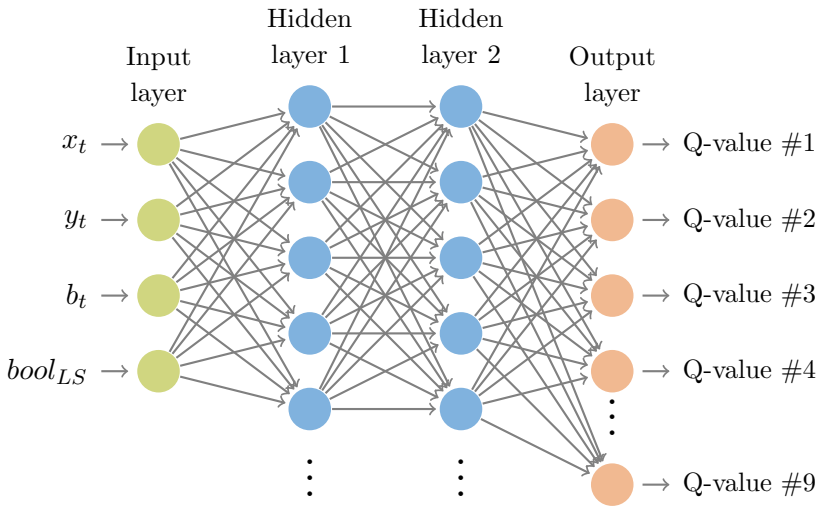


Figure 3.2 – Neural network architecture consisting of four input and nine output neurons representing the state and action space respectively, as well as two hidden and fully connected layers with 100 neurons each.

3.3.3 DQN Training Algorithm

Algorithm 1 DQN training for UAV BS trajectory planning

Initialize replay memory $\mathcal{D}$ to size D
Initialize primary network Q_θ with random parameters θ
Initialize target network $Q_{\bar{\theta}}$ with random parameters $\bar{\theta}$
1: **for** $n = 0$ to N_{max} **do**
2: Initialize state $s_0 = (x_0, y_0, b_{max}, bool_{LS})$, $t = 0$
3: **if** (n mod $N_{target} = 0$) **then**
4: $\bar{\theta} \leftarrow \theta$
5: **end if**
6: **while** $b \geq 0$ **do**
7: $a_t = \begin{cases} \text{randomly select from } \mathcal{A}, & \text{with probability } \epsilon \\ \arg\max_a Q_\theta(s_t, a), & \text{otherwise (2.15)} \end{cases}$
8: Observe r_t, s_{t+1}
9: Store $e = (s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
10: **for** $i = 0$ to m **do**
11: Sample (s_i, a_i, r_i, s_{i+1}) uniformly from $\mathcal{D}$
12: $Y_i^{\text{DQN}} = \begin{cases} r_i, & \text{if terminal} \\ r_i + \gamma \max_{a'} Q_{\bar{\theta}}(s_{i+1}, a'), & \text{otherwise according to (2.13)} \end{cases}$
13: Compute $L_i^{\text{DQN}}(\theta) = \mathbb{E}\left[\left(Q_\theta(s_i, a_i) - Y_i^{\text{DQN}}\right)^2\right]$ according to (2.12)
14: **end for**
15: $\theta \leftarrow \theta - \eta \cdot \frac{1}{m} \nabla_\theta \sum_{i=1}^{m} L_i^{\text{DQN}}(\theta)$
16: $t = t + 1$
17: **end while**
18: $\epsilon \leftarrow \epsilon_{final} + (\epsilon_{start} - \epsilon_{final}) e^{-\kappa n}$
19: **end for**

In the following, the DQN training algorithm 1 is described in more detail. After initialization of replay memory buffer and network parameters, a new learning episode is started by resetting the time index and the drone's position, as well as the drone's battery (line 2). Every N_{target} episodes, the target network parameters $\hat{\theta}$ are updated with a hard update (line 4).

As long as there is energy left in the UAV's battery, the mission continues and the agent makes a movement decision according to the ϵ-greedy policy (2.15). With probability ϵ, an action is randomly selected from the action space $\mathcal{A}$ and otherwise the action that maximizes the Q-function in the current state. Subsequently, the environment assigns a reward r_t and propagates the UAV to the next state s_{t+1}. The new experience tuple is saved in the replay memory (line 9).

To train the DQN, a minibatch of m experiences is sampled uniformly from the replay buffer $\mathcal{D}$, the target value Y_i^{DQN} is set using the target network and the loss computed according to (2.12) (lines 10-14). Using the adaptive moment estimation

(Adam) optimizer [100], an improved version of stochastic gradient descent (SGD), the primary network parameters θ are updated with learning rate η (line 15). After the battery is empty and the mission is over, the probability for random exploration of the state space is exponentially decayed with decay constant κ (line 18). Algorithm 1 terminates when the final learning episode $n = N_{max}$ is reached.

3.4 Simulation and Numerical Results

3.4.1 Simulation Setup

Algorithm 1 is applied to various aerial BS scenarios in the next sections. As depicted in Fig. 3.3a, the scenarios are defined by a grid world of size $1000\,\mathrm{m} \times 1000\,\mathrm{m}$ discretized by a cell size of $c = 100\,\mathrm{m}$ (121 unique geometric positions). The UAV's start and final position are at the origin and the upper right corner, respectively. The UAV serves $K = 10$ users. While Fig. 3.3 illustrates a simple environment with $L = 1$ LS, Fig. 3.5 and Fig. 3.6 depict scenarios with a number of $L = 2$ LSs.

As described in section 3.3.1, in addition to hovering/landing, the action space of the UAV is limited to 8 movement directions due to the geometric restrictions. With the above chosen grid world size $M = 1000\,\mathrm{m}$ and cell size $c = 100\,\mathrm{m}$, this leads to the following possible flying speeds and angles for the UAV:

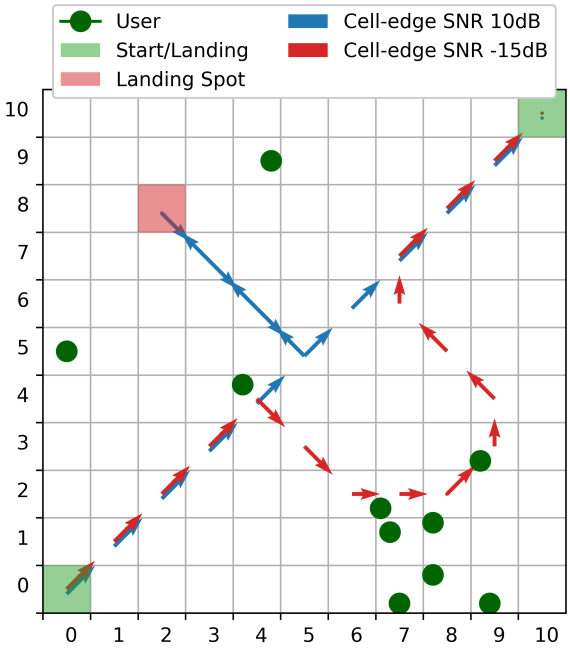
$$v \in \left\{ \begin{bmatrix} 0\,\mathrm{m\,s^{-1}} \\ 0 \end{bmatrix}, \begin{bmatrix} 12.5\,\mathrm{m\,s^{-1}} \\ \phi \end{bmatrix}, \begin{bmatrix} 17.7\,\mathrm{m\,s^{-1}} \\ \phi + \frac{\pi}{4} \end{bmatrix} \right\} \tag{3.14}$$

with $\phi \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. The UAV is assumed to travel at a constant altitude of $h = 40\,\mathrm{m}$. The drone's battery content is $b_{max} = 31$ Wh when fully charged. Power consumption for flying and communication is assumed to be $p_f = 400$ W and $p_c = 40$ W, respectively. These values reflect our experiences from real-world UAV BS experiments at EURECOM [96].

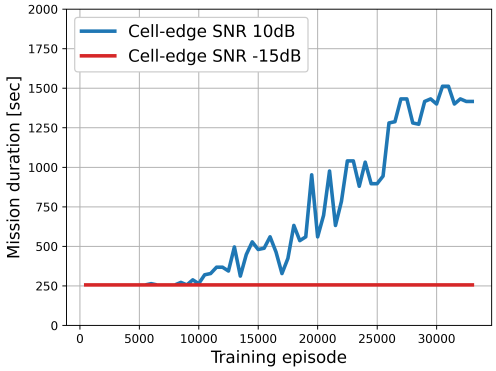
3.4.2 Scenario 1 - Different SNR Conditions

We start by investigating the basic scenario 1 as depicted in Fig. 3.3a. The $K = 10$ users are relatively unevenly distributed, with a cluster in the lower right corner. A single LS is available in the scenario and relatively far away from the user cluster. This setup is useful to show how the agent adapts to a change in wireless link conditions and that landing in the LS, although saving energy, is not always the optimal behavior for the overall data collection performance.

To this end, Fig. 3.3a shows two different trajectories under different cell-edge SNR conditions of 10db and -15dB. The cell-edge SNR is defined as the SNR of the radio link between the UAV at center position (500m, 500m) and a user maximally far apart, e.g. at (0m, 1000m). For a low cell-edge SNR, the drone ignores the LS and instead flies around the maximum sum rate point near the user cluster in the lower right corner and returns to the final position in time before its battery runs out. Under high cell-edge SNR conditions (10dB), the UAV BS learns to obtain an overall higher sum rate result by landing in the LS and extending the mission duration by conserving energy. It is also



(a) Final UAV BS trajectories.



(b) Mission duration over training episodes.

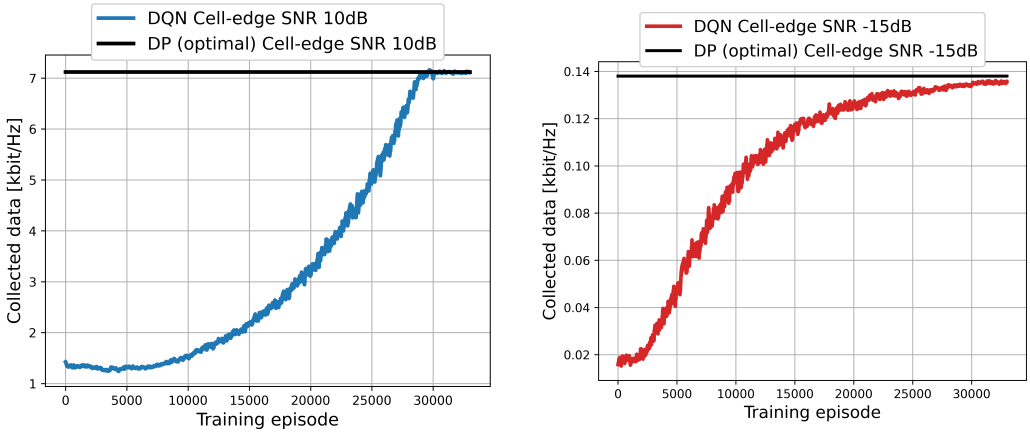
Figure 3.3 – Scenario 1 includes a single $L = 1$ landing spot. The DQN training is run under two different cell-edge SNR conditions and the final trajectories compared.

interesting to note that the UAV does not use the direct path to reach its final destination from the LS, but takes a detour towards the center of the map, bringing it closer to the user cluster.

The difference in mission duration can clearly be seen in Fig. 3.3b, as the UAV remains flying for the whole mission in the scenario with cell-edge SNR -15dB. As power consumption while flying is constant, the mission duration stays the also constant in this case.

3.4.3 Comparison with Dynamic Programming

Scenario 1 was chosen in such a way that it enables direct comparison with dynamic programming (DP) approach from [95]. DP is an optimization method based on breaking a problem into simpler sub-problems in a recursive manner. Given an initial state s_0, the optimal cost for the trajectory planning problem in scenario 1 can be computed recursively using Bellman's equations by proceeding backwards in time [101]. The solution obtained by DQN, in contrast, offers no guarantee of optimality. Empirically, from Fig, 3.5, it can be seen that the DQN converges to the maximum of collected data as well. It is important to note again that, while absolutely no prior knowledge is necessary and no modelling assumptions are made with the DQN approach, the DP approach requires perfect model knowledge or requires to make assumptions about the environment. Furthermore, the



(a) Collected data for cell-edge SNR 10dB

(b) Collected data for cell-edge SNR -15dB.

Figure 3.4 – Performance comparison of the collected data per mission obtained by the DQN approach over training episodes with the optimal trajectory obtained by DP.

DP requires large computational resources [95], that are not supportable for larger maps. In the following chapters of this thesis, we show that the deep reinforcement learning approach can adapt with reasonable computational demand to larger and more complex environments.

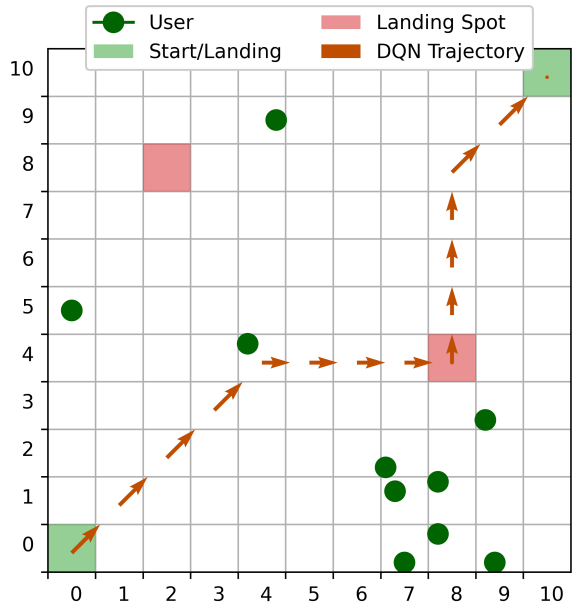
3.4.4 Scenario 2 - Decision between Landing Spots

With reference to the problem of the exploration-exploitation dilemma described in section 2.3.4, it is worth considering whether the ϵ-greedy exploration policy as used in algorithm 1, is able to explore the state-action space effectively. As depicted in Fig. 3.5a, this scenario requires that the agent decides between two LSs: one in a much more favorable location close to the user cluster and one further away. Recall that the UAV agent has absolutely no reference points apart from its own position and must build an internal representation of the environmental features such as LS positions and distance to users or final destination.

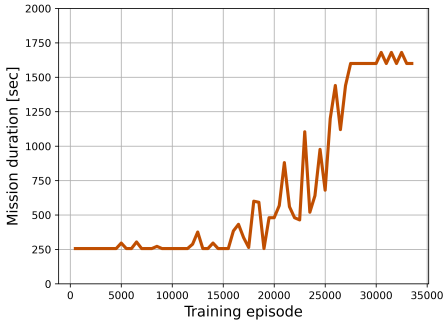
The depicted training run was specifically selected for the fact that, by virtue of random exploration, the UAV discovers the less favorable LS first and also begins to use it to extend its mission time, but continues exploring nonetheless and finally also discovers the other LS. The final trajectory as depicted, then incorporates the more favorable LS to obtain an overall higher amount of collected data over the whole mission time. This is classical case where if exploration had ended too early, the agent would have gotten stuck in a suboptimal strategy exploiting its incomplete knowledge of only one LS.

3.4.5 Scenario 3 - High and Low Shadowing Loss

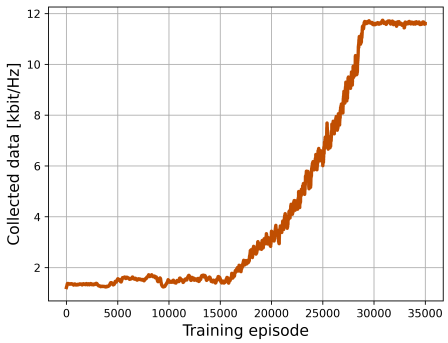
In addition to the environmental features of scenario 1 and 2 (start/final positions, user distribution), scenario 3 includes an obstacle that causes shadowing and obstructs the



(a) Final UAV BS trajectory.



(b) Mission duration over training episodes.



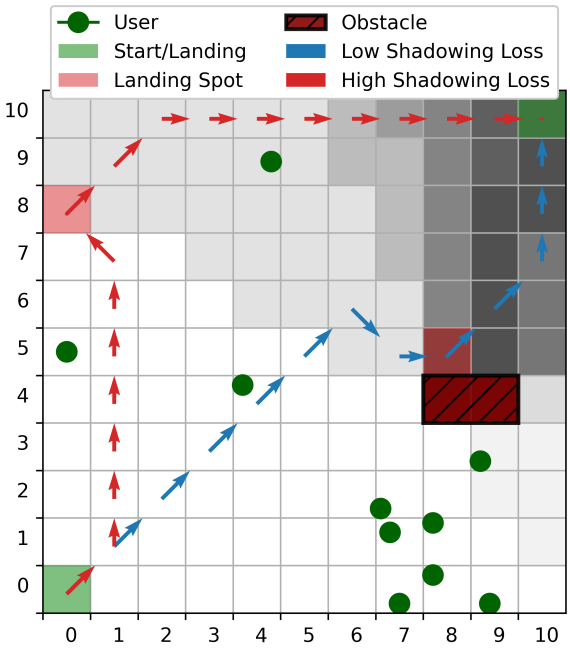
(c) Data collected over training episodes.

Figure 3.5 – Scenario 2 includes two landing spots, one favorably close to a user cluster, the other further away.

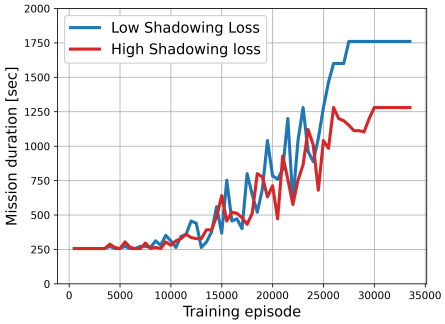
LoS connection to some users. The shadowed areas are depicted in Fig. 3.6a as different shades of gray, where the darker regions are shadowed from more users. The LS towards the right of the map is in the shadow of the obstacle whereas the left LS provides a LOS connection to all 10 users, but is further away from the user cluster.

Fig. 3.6a shows two results for high shadowing loss ($\beta = 0.01$) and low shadowing loss ($\beta = 0.1$), both under high cell-edge SNR conditions (10dB). With low shadowing, the drone stays on the direct line between start and final position until reaching the LS where it lands and conserves energy. With the minimum amount of energy left in the battery that is required to reach the final position, it restarts from the LS to arrive at the final position in time.

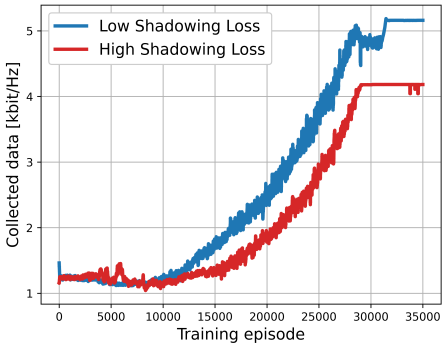
In contrast, higher shadowing loss leads the agent to realize that the left LS, despite being far away from most users and requiring more energy to reach, leads to a better overall sum rate result in the long run. Even under challenging conditions with random small-scale fading and shadowing obstacles in the environment, the DQN agent is able to discriminate between multiple LSs and different channel conditions to achieve the best long-term result. Fig. 3.6c shows the overall collected data per training episode (or completed mission) during the learning process. The overall mission duration in Fig. 3.6b for the high shadowing loss case does not reach the same level as for low shadowing



(a) Final UAV BS trajectory.



(b) Mission duration over training episodes.



(c) Data collected over training episodes.

Figure 3.6 – Scenario 3 features $L = 2$ landing spots and an obstacle that causes shadowing from some users on the map. The final results for high and low shadowing loss are compared.

loss conditions, owing to the fact that the UAV must fly a longer distance to reach the unobstructed LS.

The training converges for both shadowing conditions to a stable solution at about $n = 32,000$ episodes. Note that also in this scenario, as the agent has no access to prior information about user or obstacle positions, their positions and the resulting shadowing effect must be learned and internally represented by the agent.

3.5 Conclusion

In this chapter, we have introduced a Q-learning system that trains a neural network to make movement decisions for an autonomous UAV BS under an energy constraint. The training procedure was shown to adapt effectively to complex environmental effects like small-scale fading and obstacle shadowing. The UAV agent also learns to effectively exploit landing spots to extend its mission service time and discriminate between favorable and less favorable landing spot locations, while maximizing the sum rate of the transmission over the whole aerial BS mission. In comparison to other algorithms like dynamic

programming proposed in [95], this is possible without a model or any prior information about the environment. This is also the differentiating factor in the context of the following chapters of this thesis: we were able to find efficient UAV trajectories without making any explicit assumptions about the mission, environment or the underlying models. The price of this flexibility is the long training time and the fact that training must be rerun when parameters of the scenario change. In many cases, some additional information about the mission might be actually easily available, e.g. a map of the environment the UAV is flying in. This is the type of information that we exploit in the next chapter to generalize the learned control policy over a large mission parameter space. This means that the training procedure does not need to be rerun when certain parameters change and the UAV agent can adapt instantly to the new scenario.

Chapter 4

Multi-Scenario UAV Data Harvesting in IoT Networks

4.1 Introduction

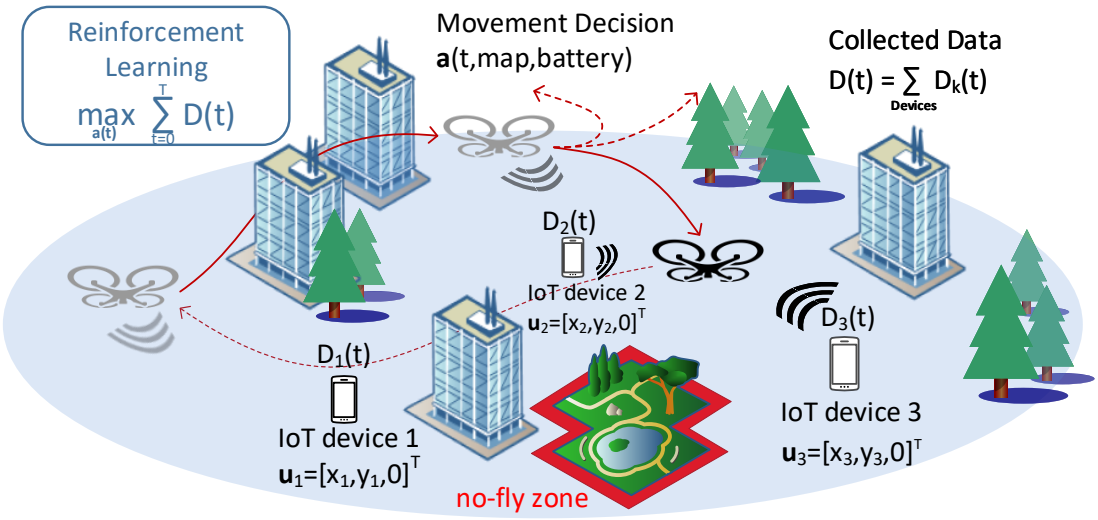


Figure 4.1 – Overview of the single-UAV data harvesting scenario. The UAV can base its movement decision on map information in addition to its own position and battery status.

In the previous chapter, we have shown that a DQN-based trajectory planning approach for aerial BSs enables us to design trajectories that reach optimal performance levels - without any assumptions about the underlying model and without any prior knowledge about the environment or the scenario. This goes hand in hand with a great demand for training data, as the UAV agent has to deduce all the information from trial-and-error exploration. Even more problematic is the fact that if something about the scenario changes, like a user position, the training procedure must be rerun again in

order to adapt to the new scenario. However, in many application cases, some additional information about the environment is actually available, e.g. a map of the environment. In this chapter, we aim to make use of map information to design a DRL algorithm that can generalize the trajectory design over multiple scenarios, i.e. take varying parameters in the scenario during training into account and find a trajectory planning policy that adapts instantly to a change in the scenario and does not require retraining. This is a much more complex problem as will become evident in the following sections. If deep RL methods are to be applied in real-world missions, the prohibitively high training data demand poses one of the most severe challenges [75]. By taking varying parameters in the design and training of the neural network model into account, we take a step towards the mitigation of this challenge.

We also move from the scenario of an aerial BS to a UAV on a data collection from distributed Internet of Things (IoT) devices mission. Fig. 4.1 provides a visualization of the mission setting. Here, the IoT devices have a finite amount of data that needs to be collected by the drone. For instance, IoT operators can deploy UAV data harvesters in the absence of otherwise expensive cellular infrastructure nearby. Another advantage is the throughput efficiency benefits related to having UAVs that describe a flight pattern that brings them close to the IoT devices. As an example in the context of infrastructure maintenance and preserving structural integrity, Hitachi is already commercially deploying partially autonomous UAVs that collect data from IoT sensors embedded in large structures, such as the San Juanico and Agas-Agas Bridges in the Philippines [8].

Collecting data from sensor devices in an urban environment imposes challenging constraints on the trajectory design for autonomous UAVs. Battery energy density restricts mission duration for quadcopter drones severely, while the complex urban environment poses challenges in obstacle avoidance and the adherence to regulatory no-fly zones (NFZs). Additionally, the wireless communication channel is characterized by frequent fluctuations in attenuation through alternating line-of-sight (LoS) and non-line-of-sight (NLoS) links. DRL approaches offer the opportunity to balance challenges and data collection goal for complex environments in a straightforward way by combining them in the reward function. Another reason for the popularity of the DRL paradigm in this context is the computational efficiency of DRL inference. DRL is also one of the few methods that allows us to tackle the complex task directly.

4.1.1 Related Work

Although quite a few previous approaches to path planning for UAVs providing some form of communication services or collecting data based on DRL existed before the work presented in this chapter, it is crucial to note that the majority of previous works concentrates on only finding the optimal trajectory solution for one set of scenario parameters at a time, requiring full or partial retraining if the scenario changes. In contrast, the presented approach aims to train and generalize over a large scenario parameter space directly, finding efficient solutions without the need for lengthy retraining, but also increasing the complexity of the path planning problem significantly.

A particular variety of IoT data collection is the one tackled in [102], where the authors propose a DQN-based solution to minimize the age of information of data collected from sensors. In contrast to our approach, the mentioned approaches are set in much simpler environments and agents have to undergo computationally expensive retraining when scenario parameters change. The authors in [67] investigate table-based Q-learning for UAV data collection. As mentioned in the previous chapter, table-based Q-learning is not suitable for the complex state-action spaces investigated here. Deep deterministic policy gradient (DDPG), an actor-critic RL method, was proposed by Qi *et al.* [103] to learn a continuous control policy for a UAV providing persistent communications coverage to a group of users in an environment without obstacles. If a critical scenario parameter like the number of users changes, the agent has to undergo computationally expensive retraining.

There are also many previous approaches to UAV data collection that are not based on RL and only find a solution for one set of scenario parameters at a time. Esrafilian *et al.* [104] proposed a two-step algorithm to optimize a UAV's trajectory and its scheduling decisions in an urban data collection mission using a combination of dynamic and sequential convex programming. While set in a similar environment, the scenario does not account for NFZs or obstacle avoidance as the drone is assumed to always fly above the highest building. This also holds for the hybrid offline-online optimization approach presented in [105], where a preliminary trajectory is computed before the UAV's start based on a probabilistic LoS channel model and then optimized while the UAV is on its mission in an online fashion.

Some works under the paradigm of mobile crowdsensing, where mobile devices are leveraged to collect data of common interest, have also suggested the use of UAVs for data collection. Liu *et al.* [66] proposed an RL multi-agent DDPG algorithm collecting data simultaneously with ground and aerial vehicles in an environment with obstacles and charging stations. While their approach also makes use of convolutional processing to exploit a map of the environment, they do not center the map on the agent's position, which we show to be highly beneficial. Furthermore, in contrast to our method, control policies have to be relearned entirely when scenario and environmental parameters change. In [106], after partitioning a sensing area without obstacles into subregions, a fixed-wing UAV is assigned to each subregion and its trajectory optimized for data collection. The authors compare two sets of non-RL algorithms that both mandate complete recomputation to adapt to changing scenario parameters.

4.1.2 Contributions

In particular, this chapter will discuss the following aspects and contributions in detail:

- Introducing a double deep Q-network (DDQN) method to control a UAV on an IoT data harvesting mission, maximizing collected data under flying time and navigation constraints without prior information about the wireless channel characteristics;
- Showing the considerable increase in learning efficiency for the RL agent when exploiting a centered multi-layer map of the environment;

- Learning to effectively adapt to variations in environmental and scenario parameters as the first step to more realistic RL methods in the context of UAV IoT data collection.

4.2 System Model and MDP

4.2.1 System Model

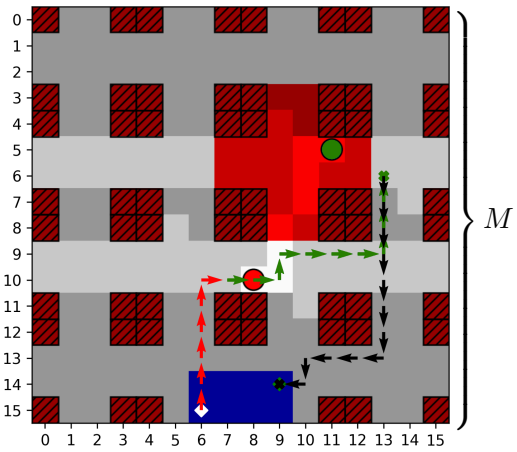


Figure 4.2 – Example of a single UAV collecting data from two IoT devices in an urban environment of size $M \times M$ with NFZs, a single start/landing zone, and buildings causing shadowing.

	Symbol	Description
DQN Input		Start and landing zone
		Regulatory no-fly zone (NFZ)
		Buildings
		IoT device
Visualization		Summation of building shadows
		Starting and landing positions during an episode
		UAV movement while comm. with green device
		Hovering while comm. with green device
		Actions without comm. (all data collected)

Table 4.2 – Legend for scenario plots.

In the following, only the key differences to the model from the previous chapter are summarized, while many assumptions are actually identical. We again consider a square grid world of size $M \times M \in \mathbb{N}^2$ with cell size c and the set of all possible positions

$\mathcal{M}$. Discretization of the environment is a necessary condition for our map-processing approach, however note that our method can be applied to any rectangular grid world. The environment contains L designated start/landing positions given by the set

$$\mathcal{L} = \left\{ \left[x_i^l, y_i^l\right]^{\mathrm{T}},\ i = 1, \ldots, L,\ : \left[x_i^l, y_i^l\right]^{\mathrm{T}} \in \mathcal{M} \right\}$$

and the combination of the Z positions the UAV cannot occupy is given by the set

$$\mathcal{Z} = \left\{ [x_i^z, y_i^z]^{\mathrm{T}},\ i = 1, \ldots, Z,\ : [x_i^z, y_i^z]^{\mathrm{T}} \in \mathcal{M} \right\}.$$

This includes buildings which the UAVs cannot fly over and regulatory no-fly zones (NFZ). The number of B obstacles blocking wireless links are given by the set

$$\mathcal{B} = \left\{ \left[x_i^b, y_i^b\right]^{\mathrm{T}},\ i = 1, \ldots, B,\ : \left[x_i^b, y_i^b\right]^{\mathrm{T}} \in \mathcal{M} \right\}.$$

The lowercase letters l, z, b indicate the coordinates of the respective set of environmental features $\mathcal{L}, \mathcal{Z}, \mathcal{B}$. An example of a grid world is depicted in Fig. 4.2, where obstacles, NFZs, start/landing zone, and an example of a UAV trajectory are marked as described in the attached legend in Tab. 4.2.

Identical to the model in section 3.2.1, the UAV's data collection mission is over at time $T \in \mathbb{N}$, where the time horizon is discretized into equal mission time slots $t \in [0, T]$. The UAV's position is given by $\mathbf{p}(t) = [x(t), y(t), z(t)]^{\mathrm{T}}$ with altitude $z(t) \in \{0, h\}$. The k-th IoT device is located on ground level at $\mathbf{u}_k = [x_k, y_k, 0]^{\mathrm{T}} \in \mathbb{R}^3$ with $k \in \mathcal{K}$. Mission time slots are chosen sufficiently small so that the UAV's velocity $v(t)$ can be considered to remain constant in one time slot. The UAV is limited to moving with constant velocity V or hovering, i.e. $v(t) \in \{0, V\}$ for all $t \in [0, T]$. The remaining flying time of the UAV $b(t) \in \mathbb{N}$ is initialized to T time steps and decremented by one after each action the UAV takes. In section 5.2.1, we briefly comment on the validity of this simplistic but for small quadcopters still realistic energy model.

As it is expected that the communication channel is subject to faster changes than the UAV's movement, we partition each mission time slot $t \in [0, T]$ into a number of $\delta \in \mathbb{N}$ communication time slots. The communication time index is then $n \in [0, N]$ with $N = \delta T$. The number of communication time slots per mission time slot δ is chosen sufficiently large so that the UAV's position, which is interpolated linearly between $\mathbf{p}(t)$ and $\mathbf{p}(t+1)$, and the channel gain can be considered constant within one communication time slot.

Similar to the channel model in [104], the communication links between UAV and the K IoT devices are modeled as LoS/NLoS point-to-point channels with log-distance path loss and shadow fading according to section 2.1.2. Note that this is different from the channel model in chapter 3, as there direct comparison with another approach was necessary. We now follow equations (2.2) and (2.1) directly.

The sensor nodes are served by the UAV in a simple time-division multiple access (TDMA) manner where, in each communication time slot $n \in [0, N]$, the sensor node $k \in [1, K]$ with the highest $\mathrm{SNR}_k(n)$ with remaining data to be uploaded is picked by the

scheduling algorithm. The TDMA constraint for the scheduling variable $q_k(n) \in \{0, 1\}$ is given by

$$\sum_{k=1}^{K} q_k(n) \leq 1, \ n \in [0, N]\,. \tag{4.1}$$

The achievable throughput for one mission time slot t is then the sum of the achieved rates of the corresponding communication time slots $n \in [\delta t, \delta(t+1) - 1]$ over K sensor nodes and given by

$$C(t) = \sum_{n=\delta t}^{\delta(t+1)-1} \sum_{k=1}^{K} q_k(n) R_k(n). \tag{4.2}$$

The central goal of the trajectory optimization problem is the maximization of throughput over the whole data collection mission while minimizing flight duration, subject to the constraints of maximum flight time, adherence to NFZs, obstacle avoidance, and safe landing in designated landing areas. We translate this optimization problem again into a reward function as part of a Markov decision process.

4.2.2 Markov Decision Process

The state at mission time t in the grid world of size $M \times M$ is given by $s_t = (\mathbf{D}_t, \mathbf{p}_t, b_t, \mathbf{M}, \mathbf{U})$ and consists of five components:

- $\mathbf{D}_t \in \mathbb{R}^{K \times 2}$ represents the initially available and the already collected data for each device;
- $\mathbf{p}_t \in \mathbb{R}^2$ is the UAV position projected on the ground;
- $b_t \in \mathbb{N}$ is the UAV's remaining flying time;
- $\mathbf{M} \in \mathbb{B}^{M \times M \times 3}$ is the map of the physical environment in the Boolean domain $\{0, 1\}$ encoded with three map layers for start/landing positions, NFZs and buildings;
- $\mathbf{U} \in \mathbb{R}^{K \times 2}$ are the 2D coordinates of the K IoT devices.

Note that the state is transformed before being fed into the agent as detailed in 4.3. Considering the five described components, the total size of the state space is

$$\mathcal{S} = \underbrace{\mathbb{R}^2}_{\substack{\text{Position}}} \times \underbrace{\mathbb{B}^{M \times M \times 3}}_{\substack{\text{Environment} \\ \text{Map}}} \times \underbrace{\mathbb{R}^{K \times 2}}_{\substack{\text{Device} \\ \text{Positions}}} \times \underbrace{\mathbb{R}^{K \times 2}}_{\substack{\text{Device} \\ \text{Data}}} \times \underbrace{\mathbb{N}}_{\substack{\text{Flying} \\ \text{Time}}}.$$

In contrast to chapter 3, we simplify the action space as the additional diagonal movement directions do not improve the fundamental trajectory planning strategy, but add implementation complexity due to the resulting different traveling speeds. It assumed that the UAV would anyway fly at the highest economical speed or hover if no movement action would improve its current communication performance. We however differentiate between hovering and landing explicitly now, as the UAV can also decide to end its

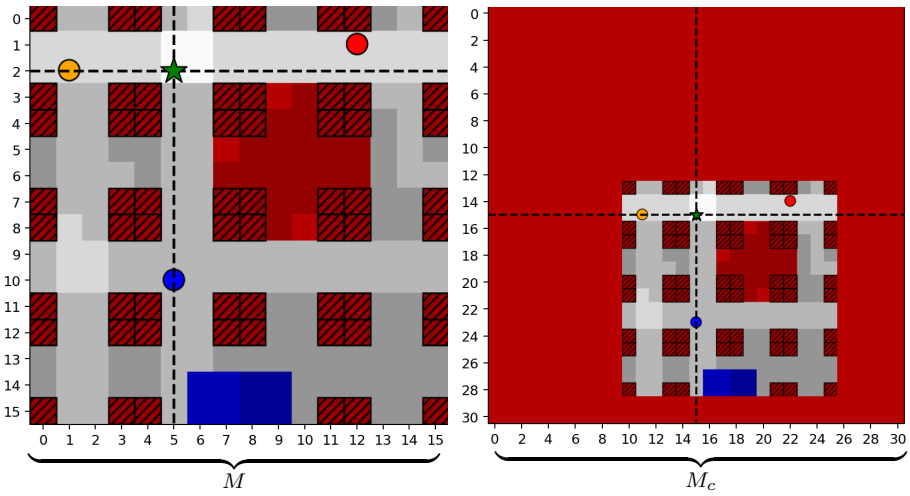
mission early with energy left in its battery by landing. The UAV is limited to six actions contained in the action space

$$\mathcal{A} = \left\{ \underbrace{\begin{bmatrix}0\\0\\0\end{bmatrix}}_{\text{hover}}, \underbrace{\begin{bmatrix}c\\0\\0\end{bmatrix}}_{\text{east}}, \underbrace{\begin{bmatrix}0\\c\\0\end{bmatrix}}_{\text{north}}, \underbrace{\begin{bmatrix}-c\\0\\0\end{bmatrix}}_{\text{west}}, \underbrace{\begin{bmatrix}0\\-c\\0\end{bmatrix}}_{\text{south}}, \underbrace{\begin{bmatrix}0\\0\\-h\end{bmatrix}}_{\text{land}} \right\}. \tag{4.3}$$

The reward function maps state-action pairs to a real-valued reward, i.e. $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Representing the mission goals, the reward function consists of the following components:

- r_{data} *(positive)* the data collection reward given by the achieved throughput (4.2) in the current time slot;
- r_{sc} *(negative)* safety controller (SC) penalty in case the drone has to be prevented from colliding with a building or entering an NFZ;
- r_{mov} *(negative)* constant movement penalty that is applied for every action the UAV takes without completing the mission;
- r_{crash} *(negative)* penalty in case the drone's remaining flying time reaches zero without having landed safely in a landing zone.

4.3 Map Processing



(a) Non-centered input map (b) Centered input map

Figure 4.3 – Comparison of non-centered and centered input maps, with UAV position represented by the green star and the intersection of the dashed lines.

As it is realistic to assume that map information is available for an IoT network in an urban environment, this section introduces the necessary map processing. However, we only offer a short summary here and would like to refer to the following chapter 5 for a more rigorous formulation. The global map is composed of the static environmental map and a dynamic device data map, which is formatted as two real-valued map layers. The first layer represents the data available for collection from each device at its respective position and the second layer records the data that has already been collected throughout the mission.

With this encoding, it would be possible to feed the map data directly into the agent as it was done in [78], with an input space defined through

$$\mathcal{I} = \underbrace{\mathbb{R}^2}_{\substack{\text{Position}}} \times \underbrace{\mathbb{B}^{M\times M\times 3}}_{\substack{\text{Environment}\\ \text{Map}}} \times \underbrace{\mathbb{R}^{M\times M\times 2}}_{\substack{\text{Device Data}\\ \text{Map}}} \times \underbrace{\mathbb{N}}_{\substack{\text{Flying}\\ \text{Time}}}.$$

However, here we show that centering the map layers on the UAV's position greatly benefits its ability to generalize over varying scenario parameters. While centering an input map was already applied to local maps that only show the area immediately surrounding the agent, such as in the related field of UAV navigation [107], it is here applied for the first time to global maps in a UAV data collection scenario.

The map centering process inside the computational graph is illustrated in Fig. 4.3 with a legend provided in Table 4.2. For centering, the maps are expanded to $(2M-1)\times(2M-1)$ in order to enable the agent to observe the entire map independent of its position in it. Translation of the original map centers the expanded map on the UAV's position. The resulting input space is defined through

$$\mathcal{I}_c = \underbrace{\mathbb{B}^{(2M-1)\times(2M-1)\times 3}}_{\substack{\text{Centered Environment}\\ \text{Map}}} \times \underbrace{\mathbb{R}^{(2M-1)\times(2M-1)\times 2}}_{\substack{\text{Centered Device}\\ \text{Data Map}}} \times \underbrace{\mathbb{N}}_{\substack{\text{Flying}\\ \text{Time}}}.$$

4.4 Extensions to the DQN Paradigm

In section 2.3.3, we have introduced the DQN paradigm as introduced by Mnih et al. [91]. Here we make use of additional improvements to the training process that were suggested in [108], resulting in the inception of *double deep Q-networks (DDQNs)*. With the application of this extension, we avoid the overestimation of action values under certain conditions in standard DQN and arrive at the loss function for our network given by

$$L^{\text{DDQN}}(\theta) = \mathbb{E}_{s,a,s'\sim\mathcal{D}}[(Q_\theta(s,a) - Y(s,a,s'))^2] \tag{4.4}$$

where the target value is given by

$$Y^{\text{DDQN}}(s,a,s') = r(s,a) + \gamma Q_{\bar{\theta}}(s', \arg\max_{a'} Q_\theta(s',a')). \tag{4.5}$$

In the standard DQN paradigm, new experiences made by the agent, represented by quadruples of (s,a,r,s'), are stored in the replay memory $\mathcal{D}$. During training, a minibatch

Parameter	Value	Description
$\|\mathcal{D}\|$	50,000	replay memory buffer size
N_{max}	10,000	maximum number of training episodes
β	0.1	temperature parameter (2.16)
m	128	minibatch size
γ	0.95	discount factor for target value in (4.5)
τ	0.005	target network update factor (2.14)

Table 4.3 – Hyperparameters for DDQN training with centered map input.

of size m is sampled uniformly from $\mathcal{D}$ and used to compute the loss. The size of the replay memory $|\mathcal{D}|$ was shown to be an essential hyperparameter for the agent's learning performance and typically must be carefully tuned for different tasks or scenarios. Zhang and Sutton [109] proposed *combined experience replay* as a remedy for this sensitivity with very low computational complexity $\mathcal{O}(1)$. In this extension to the replay memory method, only $m-1$ samples of the minibatch are sampled from memory, and the latest experience the agent made is always added. This corrected minibatch is then used to train the agent. Therefore, all new transitions influence the agent immediately, making the agent less sensitive to the selection of the replay buffer size in our approach. In this section we apply DDQN and combined experience replay to the UAV trajectory planning problem.

4.5 Neural Network Model

Fig. 4.4 shows the DQN structure and the map centering pre-processing. Note that the device information is encoded in channel 4 and 5, but is visualized using colors from the first 3 channels. The centered map is fed through convolutional layers with ReLU activation and then flattened and concatenated with the scalar input indicating remaining flight time. After passing through fully connected layers with ReLU activation, the data reaches the last fully-connected layer of size $|\mathcal{A}|$ and without activation function, directly representing the Q-values for each action given the input state. The arg max of the Q-values, i.e. the greedy policy, is given by equation (2.15). It is deterministic and used when evaluating the agent. During training, the soft-max policy (2.16) is used. Hyperparameters for DDQN training are identical to [78] and can be found in Tab. 4.3.

4.6 Simulations

4.6.1 Simulation Setup

The UAV starts each new mission from a random position inside the start/landing zone in a world discretized into 16×16 cells where each grid cell is of size 10m $\times$ 10m. It starts with a remaining flying time of T steps, which is decremented by one after every

Figure 4.4 – DDQN architecture with map centering, with the device map encoded in separate layers, but visualized in RGB channels.

action the agent takes, no matter if moving or hovering. The UAV flies at a constant altitude of $h = 10$m inside 'urban canyons' through a city environment or open fields and is, for regulatory reasons, not allowed to fly over buildings, enter NFZs, or leave the 16×16 grid.

In this work, we aim to provide an algorithm that is able to generalize the learned UAV control policy over a large parameter space that defines the specific data collection scenario. That means that at the start of a new training episode, a set of scenario parameters is sampled randomly from a given range of possible values defining the mission. Then the mission starts and the agents are deployed to collect as much data as possible in the given circumstances. Specifically, we define a new mission through the following randomly varying scenario parameters:

- Number and position of IoT sensor nodes;
- Amount of data to be collected from IoT nodes;
- Flying time available for UAVs at mission start;
- UAV start positions.

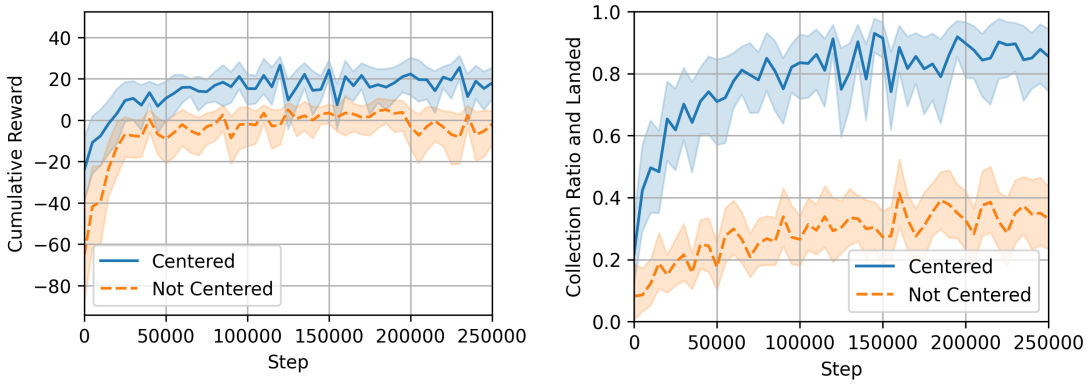
Each mission time slot contains $\delta = 4$ scheduled communication time slots. Propagation parameters (see 2.1.2) are chosen in-line with [110] according to the urban micro scenario with $\alpha_{\text{LoS}} = 2.27$, $\alpha_{\text{NLoS}} = 3.64$, $\sigma^2_{\text{LoS}} = 2$ and $\sigma^2_{\text{NLoS}} = 5$. The shadowing maps to simulate the environment were computed using ray tracing from and to the center points of cells. Transmission and noise powers are normalized through the definition of a cell-edge SNR of -15dB. The agent has absolutely no prior knowledge of the shadowing maps or wireless channel characteristics.

We use the following metrics to evaluate the agent's performance in different scenarios and to compare training instances:

- *Cumulative reward*: the sum of all rewards received throughout an episode;
- *Has landed*: records whether the agent landed in time at the end of an episode;
- *Collection ratio*: the ratio of collected data to total initially available device data at the end of a mission;
- *Collection ratio and landed*: the product of *has landed* and *collection ratio* per episode.

In contrast to many other works in deep RL, we only use *cumulative reward* as a metric to illustrate the training process and focus on *collection ratio and landed* as the main performance metric because it is most indicative of a successful UAV data collection mission.

As we train a single agent to generalize over a large scenario parameter space, defined by the number of IoT devices, position of IoT devices, data to be collected at each device, start positions, and flying time limits, evaluation is challenging. During training, we evaluate the agent's training progress in a randomly selected scenario every ten episodes



(a) Episodic cumulative reward

(b) Collection ratio and landed

Figure 4.5 – Training process comparison between centered and non-centered map input showing the average and 99% quantiles of three training processes each, with episodic metrics grouped in bins of 5000 step width.

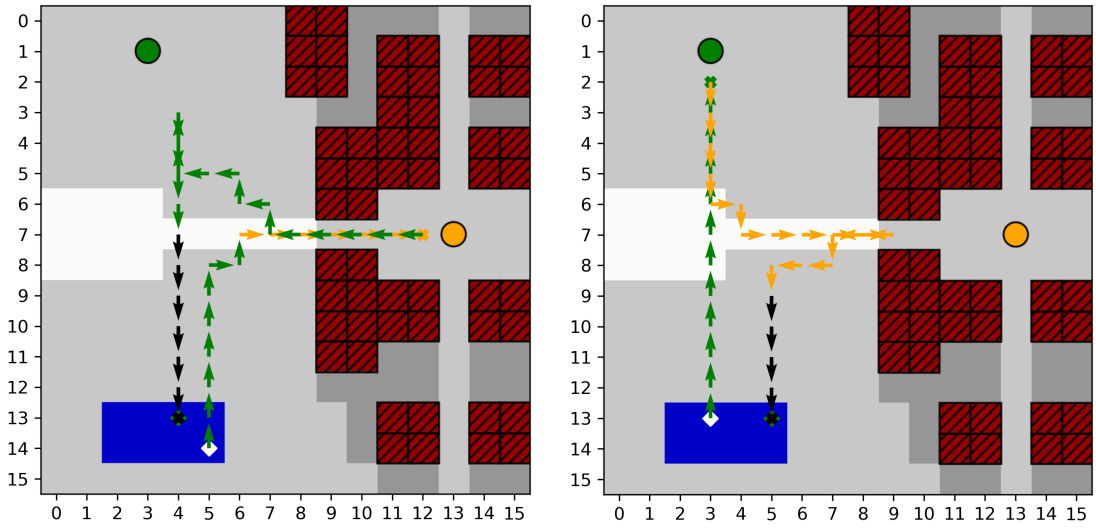
and form an average over multiple evaluations. A single evaluation could be tainted by unusually easy conditions, e.g., when all devices are placed very close to each other by chance. Therefore, only an average over multiple evaluations can be indicative of the agent's progress. As it is computationally infeasible to evaluate the trained agent on all possible scenario variations, we perform Monte Carlo analysis on a large number of randomly selected scenario parameter combinations.

4.6.2 Centered vs. Non-Centered Map

As described in Section 4.3, the map of the environment is processed to be centered on the UAV's position before feeding it into the convolutional layers of the agent. This proved to be highly beneficial to the learning performance and the generalization ability of the DDQN agent. Fig. 4.5 shows comparisons of two performance metrics, cumulative reward per episode, and achieved data collection ratio in missions with in-time landing over training time, for centered and non-centered map inputs in identical scenarios.

The difference between the agents is that the non-centered agent's convolutional layers are padded, while the centered agent's are not. By setting the number of convolutional layers to four, the last convolutional layer of the two agents have similar shape (15x15x16 for centered and 16x16x16 for non-centered), yielding a similar size flatten layer. This is done to eliminate the size factor of the flatten layer as a parameter for their performance. The only difference between the agents is that the non-centered agent receives the position as a 2D-one-hot encoded map layer similar to [78]. Each graph is averaged over three training runs to account for possible random variations in the training process. A clear performance advantage for the agent using the centered map input can be seen throughout the whole learning process.

The benefit of using a centered map is the result of a change in position to which a neuron of the "Flatten" layer (see Fig. 4.4) corresponds. If the map is not centered, the



(a) Equal data amounts

(b) Orange device with a quarter of the green device's data

Figure 4.6 – Illustration of the same agent adapting to differences in collectible data with all other mission parameters fixed.

neurons in that layer correspond to features at *absolute* positions. If the map is centered, they correspond to features at positions *relative* to the agent. Since the agent's actions are solely based on its relative position to features, e.g. its distance to devices, learning efficiency increases considerably.

4.6.3 Collectible Data and Device Accessibility

The scenario map in Fig. 4.6 is divided into an open field and an adjacent city. To show the agent's responsiveness to differences in collectible data at the same devices, we fixed the number of IoT devices to $K = 2$, while allowing for fully randomized device positions in unoccupied map space, for each device randomized collectible data ($D_0 \in [1.0, 25.0]$ data units), randomized flying time limits ($b_0 \in [35, 70]$ steps) and eight possible start positions.

Fig. 4.6 shows the agent adapting to a change in collectible data at the two devices. The agent only enters the hard to navigate courtyard if the amount of data at the orange device requires it. While starting to communicate with the unobstructed green device in Fig. 4.6a, the agent proceeds to collect data from the harder-to-access orange device first, then picking up the rest from the green device before returning straight to the landing area. For the case in Fig. 4.6b, the UAV changes its strategy. While immediately reducing its distance to the green node after starting and collecting all its data, it collects the data from the orange device on the way back with a detour only as long as required, minimizing the overall mission duration. The UAV is also clearly able to identify unobstructed positions to communicate with the orange device.

4.6.4 Manhattan Scenario

Metric	Manhattan	Open Field and City
Has Landed	99.5%	99.9%
Collection Ratio	94.8%	90.0%
Collection Ratio and Landed	94.6%	89.9%

Table 4.4 – Performance metrics averaged over 1000 random scenario Monte Carlo iterations.

The main scenario we investigate is defined by a Manhattan-like city structure (see Fig. 4.7) containing regularly distributed city blocks with streets in between, as well as an NFZ district. In this challenging setting, we want to demonstrate the agent's ability to generalize over significant variations in scenario parameters with randomly changing device count ($K \in [2, 5]$), device data ($D_0 \in [5.0, 20.0]$ data units), maximum flying time ($b_0 \in [35, 70]$ steps), and eight possible starting positions. Similar to the previous scenario, device positions are randomized throughout the unoccupied map space.

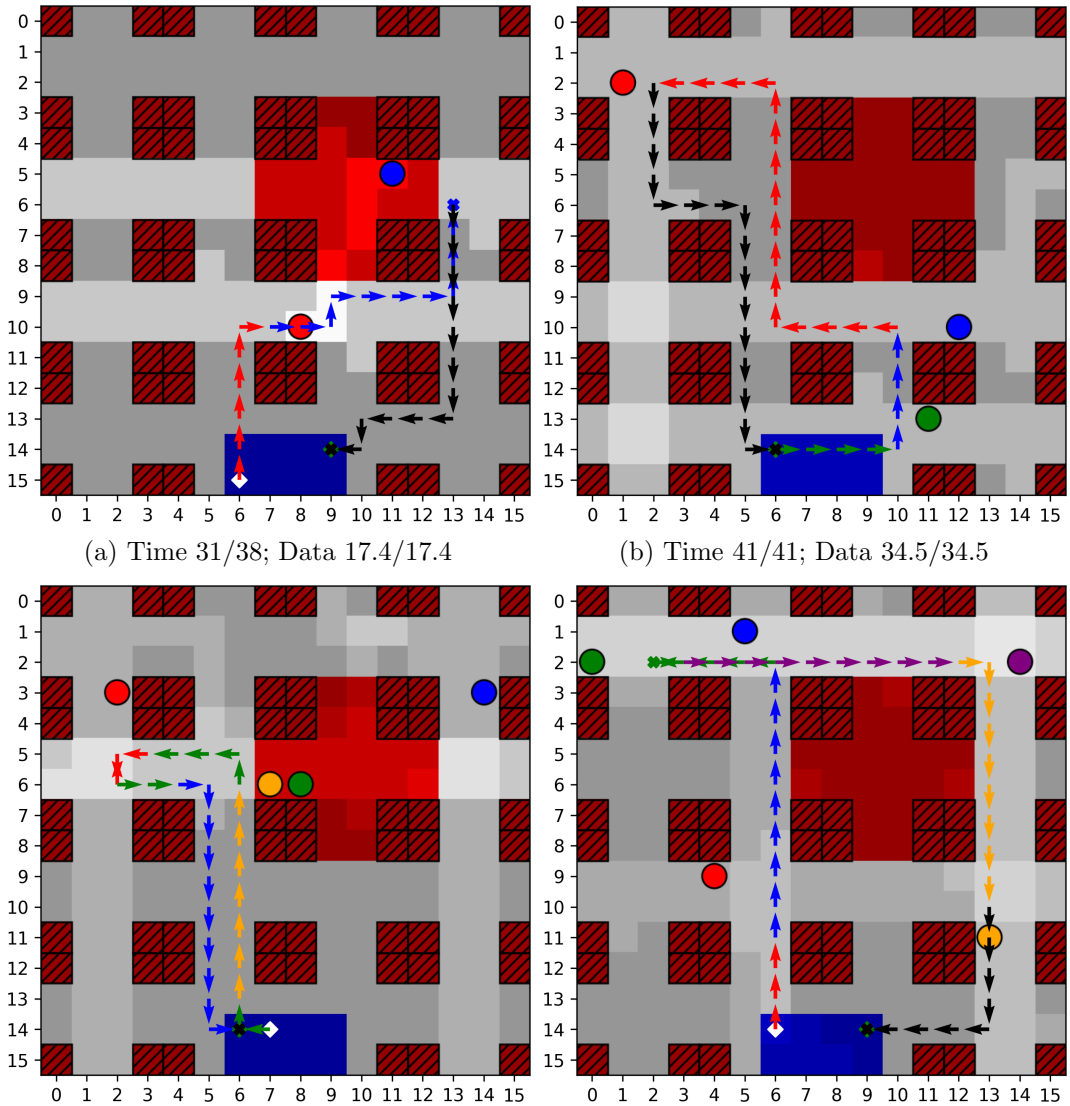
This and the previous scenario are evaluated using Monte Carlo simulations on their full range of scenario parameters with average performance metrics shown in Table 4.4. Both agents show a similarly high successful landing performance. It is expected that the collection ratio must be less than 100% in some scenario instances depending on the randomly assigned maximum flying time and IoT device parameters.

In Fig. 4.7, four scenario instances chosen from the random Monte Carlo evaluation for device counts of $K \in \{2, 3, 4, 5\}$ for 4.7a through 4.7d illustrate the agent's adaptability. With $K = 2$ devices in Fig. 4.7a, finding a trajectory is complicated by the location of the blue device inside the NFZ and the resulting shadowing effects, which have to be deduced by the agent from building and device positions. In Fig. 4.7b, the considerable distance to the red device requires the agent to exhaust its entire flight time. For the scenario in Fig. 4.7c the available flying time $T = 35$ is not sufficient to collect all data. Therefore, the agent ignores the isolated blue device and lands early after collecting all data within reach. In Fig. 4.7d the agent successfully collects all data in an efficient order while minimizing its flying time, e.g. by turning away from the green device before transmitting all its data. We observed that rerunning the same scenario configuration leads to a variation in trajectories which adapt to effects of the random communication channel fading.

4.7 Conclusion

In this chapter, we introduced a new DDQN method with combined experience replay for UAV trajectory planning in an IoT data harvesting scenario. By leveraging a neural network model that exploits information about the environment from centered map layers through convolutional processing, we showed that the UAV agent learns to effectively adapt to significant variations in the scenario such as number and position of IoT devices,

amount of collectible data or maximum flying time, without the need for expensive retraining or recollection of training data. Using this method, we have shown that the UAV balances the goals of data collection, obstacle avoidance, and minimizing mission time effectively, while not requiring any prior information about the challenging wireless channel characteristics in an urban environment. However, the presented map centering approach also introduces a new challenge, namely the linear increase of trainable parameters in the flatten layer of the neural network model with map area. In the next chapter, we will tackle this issue of scalability to larger maps. Furthermore, we have so far only looked at single UAV scenarios and the extension to the multi-UAV case brings additional challenges in terms of coordination and multi-agent reinforcement learning (MARL), which we will also detail in the following chapter. We will further extend the scenario parameter generalization of the learned control policy to a larger scenario space, including the number of deployed UAVs, that enables us to identify system-level trade-offs in the IoT data harvesting system easily.



(a) Time 31/38; Data 17.4/17.4

(b) Time 41/41; Data 34.5/34.5

(c) Time 30/35; Data 32.0/50.6

(d) Time 45/65; Data 60.7/60.7

Figure 4.7 – Illustration of the same agent adapting to differences in device count and device placement as well as flight time limits, showing used and available flying time and collected and available total data in the Manhattan scenario.

Chapter 5

Multi-UAV Coordination in Multi-Scenario Data Harvesting

5.1 Introduction

In this chapter, we extend and improve on the results from chapter 4 while considering a similar UAV-aided communications scenario with data collection from distributed IoT devices. It is reasonable to assume that a network operator deploying UAV data harvesters as part of its IoT network infrastructure, would try to increase the effectiveness of the system by deploying multiple of these UAVs. From the trajectory planning perspective, this requires taking into account the coordination aspect and a reformulation of the approach to multi-agent reinforcement learning (MARL). Hence, we formulate the path planning problem for a cooperative, non-communicating, and homogeneous team of UAVs tasked with maximizing collected data from distributed IoT sensor nodes subject to flying time and collision avoidance constraints. The path planning problem is translated into a decentralized partially observable Markov decision process (Dec-POMDP), which we solve through a DDQN approach, approximating the optimal UAV control policy without prior knowledge of the challenging wireless channel characteristics in dense urban environments.

The second improvement is related to the fact that the map centering approach as introduced previously, although leading to high learning performance gains, also increases the number of trainable neural network parameters that makes it unsustainable for larger, more realistic and complex maps. In this chapter, we also modify the scenario to differentiate between tall buildings that act as navigation obstacles, as well as smaller buildings that can be flown over by the UAV. We tackle the issue of increasing trainable parameters by exploiting a combination of centered global and local map representations of the environment that are fed into convolutional layers of the agents. We show that our proposed network architecture enables the agents to cooperate effectively by carefully dividing the data collection task among themselves, adapt to large complex environments and state spaces, and make movement decisions that balance data collection goals, flight-time efficiency, and navigation constraints.

Finally, in this work, we focus on controlling a team of UAVs, consisting of a variable number of identical drones tasked with collecting varying amounts of data from a variable number of stationary IoT sensor devices at variable locations in an urban environment. This imposes challenging constraints on the trajectory design for autonomous UAVs. Learning a control policy that generalizes over the scenario parameter space enables us to analyze the influence of individual parameters on collection performance and provide some intuition about system-level benefits.

5.1.1 Related Work

The focus of this section is to extend on the discussed literature in the previous chapter by concentrating on multi-UAV DRL trajectory planning in UAV-aided communications. Research into UAV-aided data collection from IoT devices or wireless sensors include the works [33, 66, 67, 111, 112], with [102, 113–115] concentrating on minimizing the age of information of the collected data specifically. The most relevant surveys for multi-UAV systems include [74] that spans the various application areas for multi-UAV systems from a cyber-physical perspective. The general challenges and opportunities of multi-UAV communications are also summarized in publications by Zeng *et al.* [1] and Saad *et al.* [36], which both include data collection from IoT devices. This specific scenario is also included in [52] and [61], surveys that comprise information on the classification of UAV communication applications including a focus on MARL methods.

Multi-UAV path planning for serving ground users employing table-based Q-learning is investigated in [73], based on a relatively complex 3-step algorithm consisting of grouping the users with a genetic algorithm, then deployment and movement design in two separated instances of Q-learning. The investigated optimization problem is proven to be NP-hard, with Q-learning being confirmed as a useful tool to solve it. Pan *et al.* [111] investigate an instance of multi-UAV data collection from sensor nodes formulated as a classical traveling salesman problem without modeling the communication phase between UAV and node. The UAVs' trajectories are designed with a genetic algorithm that uses some aspects of DRL, namely training a deep neural network and experience replay. In contrast to the multi-stage optimization algorithms in [73] and [111], the approach presented in this chapter consists of a more straightforward end-to-end DRL approach that scales to large and complex environments, generalizing over varying scenario parameters.

The combination of DRL and multi-UAV control has been studied previously in various scenarios. The authors in [113] focus on trajectory design for minimizing the age of information of sensing data generated by multiple UAVs themselves where the data can be either transmitted to terrestrial base stations or mobile cellular devices. Their focus lies on balancing the UAV sensing and transmission protocol in an unobstructed environment for one set of scenario parameters at a time. Other MARL path planning approaches to minimize the age of information of collected data include [114] and [115]. In [116], a swarm of UAVs on a target detection and tracking mission in an unknown environment is controlled through a distributed DQN approach. While the authors also use convolutional processing to feed map information to the agents, the map is initially unknown and has to be explored to detect the targets. The agents' goal is to learn transferable knowledge that

enables adaptation to new scenarios with fast relearning, compared to our approach to learn a control policy that generalizes over scenario parameters and requires no relearning.

Hu *et al.* [30] proposed a distributed multi-UAV meta-learning approach to control a group of drone base stations serving ground users with random uplink access demands. While meta-learning allows them to reduce the number of training episodes needed to adapt to a new unseen uplink demand scenario, several hundred are still required. The approach presented in this chapter focuses on training directly on random but observable scenario parameters within a given value range, therefore not requiring retraining to adapt. Due to the small and obstruction-less environment, no maps are required in [30] and navigation constraints are omitted by keeping the UAVs at dedicated altitudes. In [33], multi-agent deep Q-learning is used to optimize trajectories and resource assignment of UAVs that collect data from pre-defined clusters of IoT devices and provide power wirelessly to them. The focus here is on maximizing minimum throughput in a wirelessly powered network without a complex environment and navigation constraints, only for a single scenario at a time. Similarly, in [112] there is also a strong focus on the energy supply of IoT devices through backscatter communications when a team of UAVs collects their data. The authors propose a multi-agent approach that relies on the definition of ambiguous boundaries between clusters of sensors. The scenario is set in a simple, unobstructed environment, not requiring maps or adherence to multiple navigation constraints, but requiring retraining when scenario parameters change.

In [42], a group of interconnected UAVs is tasked with providing long-term communication coverage to ground users cooperatively. While the authors also formulate a POMDP that they solve by a DRL variant, there is no need for map information or processing. The scenario is set in a simple environment without obstacles or other navigation constraints. This work was extended under the paradigm of mobile crowdsensing, where mobile devices are leveraged to collect data of common interest in [66]. The authors proposed a heterogeneous multi-agent DRL algorithm collecting data simultaneously with ground and aerial vehicles in an environment with obstacles and charging stations. While in this work, the authors also suggest a convolutional neural network to exploit a map of the environment, the small grid world does not necessitate extensive map processing. Furthermore, they do not center the map on the agent's position, which is highly beneficial as discussed in the previous chapter 4. In contrast to the method presented here, control policies have to be relearned entirely in a lengthy training process for both mentioned approaches when scenario and environmental parameters change.

5.1.2 Contributions

As introduced in chapter 4, if DRL methods are to be applied in any real-world mission, the prohibitively high training data demand poses one of the most severe challenges [75]. This is exacerbated by the fact that even minor changes in the scenario, such as in the number or location of sensor devices in data collection missions, typically requires repeating the full training procedure of the DRL agent. This is the case for existing approaches such as [29, 33, 42, 66, 112–115]. Other approaches to reduce the training data demand include meta-learning [30] and transfer learning [75]. The approach introduced

here is, to the best of the author's knowledge, the first work that addresses this problem in path planning for multi-UAV data harvesting by proposing a DRL method that is able to generalize over a large space of scenario parameters in complex urban environments without prior knowledge of wireless channel characteristics based on centered global-local map processing. The main contributions in this chapter can be summarized as follows:

- We formulate a flying time constrained multi-UAV path planning problem to maximize harvested data from IoT sensors. We consider its translation to a decentralized partially observable Markov decision process (Dec-POMDP) with full reward function description in large, complex, and realistic environments that include no-fly zones, buildings that block wireless links (some possible to be flown over, some not), and dedicated start/landing zones.

- To solve the Dec-POMDP under navigation constraints without any prior knowledge of the urban environment's challenging wireless propagation conditions, we employ deep multi-agent reinforcement learning with centralized training and decentralized execution (CTDE).

- We show the advantage in learning and adaptation efficiency to large maps and state spaces through a dual global-local map approach with map centering over more conventional scalar neural network input in a multi-UAV setting.

- We extend on the feature of the proposed algorithm introduced in the previous chapter, *parameter generalization*, which means that the learned control policy can be reused over a wide array of scenario parameters, including the number of deployed UAVs, variable start positions, maximum flying times, and number, location and data amount of IoT sensor devices, without the need to restart the training procedure as typically required by other DRL approaches.

- Learning a generalized control policy enables us to compare performance over a large scenario parameter space directly. We analyze the influence of individual parameters on collection performance and provide some intuition about system-level benefits.

5.2 System Model

The system model is very similar to the one presented in section 4.2.1, but needs to be partially reformulated to account for the presence of multiple UAVs. We still consider a square grid world of size $M \times M \in \mathbb{N}^2$ with cell size c and the set of all possible positions $\mathcal{M}$. The environment contains L designated start/landing positions given by the set $\mathcal{L}$. The combination of the Z positions the UAVs cannot occupy which is given by the set $\mathcal{Z}$ which includes tall buildings which the UAVs can not fly over and regulatory no-fly zones (NFZ). The number of B obstacles blocking wireless links are given by the set $\mathbf{B}$ representing all buildings, also smaller ones that can be flown over. The exact set definitions are given in section 4.2.1.

5.2.1 UAV Model

The set $\mathcal{I}$ of I deployed UAVs moves within the limits of the grid world $\mathcal{M}$. The state of the i-th UAV is described through its:

- position $\mathbf{p}_i(t) = [x_i(t), y_i(t), z_i(t)]^{\mathrm{T}} \in \mathbb{R}^3$ with altitude $z_i(t) \in \{0, h\}$, either at ground level or in constant altitude h;
- operational status $\phi_i(t) \in \{0, 1\}$, either inactive or active;
- battery energy level $b_i(t) \in \mathbb{N}$.

Note that the assumption of all UAVs sharing the same flying altitude is not too restrictive and that the presented method allows each UAV to fly at a different altitude as long as it remains constant throughout the mission. The UAV agent's altitude can be made observable by simply adding it to the observation space along the flying time. Here, we only tackle 2D trajectory optimization, as the environment is dominated by high-rise buildings that would require long climbing phases to be overflown. The mission time limited by the UAVs' on-board batteries restricts the effectiveness of 3D control for the data collection performance given that climbing flight consumes more energy [81] and that the UAVs needs to land at ground level at the end of the mission. The data collection mission is over after $T \in \mathbb{N}$ mission time steps for all UAVs, where the time horizon is discretized into equal mission time slots $t \in [0, T]$ of length δ_t seconds. The action space of each UAV is identical to (4.3). However, each UAV's movement actions $\mathbf{a}_i(t) \in \tilde{\mathcal{A}}(\mathbf{p}_i(t))$ are limited to

$$\tilde{\mathcal{A}}(\mathbf{p}_i(t)) = \begin{cases} \mathcal{A}, & \mathbf{p}_i(t) \in \mathcal{L} \\ \mathcal{A} \setminus [0, 0, -h]^{\mathrm{T}}, & \text{otherwise}, \end{cases} \tag{5.1}$$

where $\tilde{\mathcal{A}}$ defines the set of feasible actions depending on the respective UAV's position, specifically that the landing action is only allowed if the UAV is in the landing zone.

The distance the UAV travels within one time slot is equivalent to the cell size c. Mission time slots are chosen sufficiently small so that each UAV's velocity $v_i(t)$ can be considered to remain constant in one time slot. The UAVs are limited to moving with horizontal velocity $V = c/\delta_t$ or standing still, i.e. $v_i(t) \in \{0, V\}$ for all $t \in [0, T]$. Each UAV's position evolves according to the motion model given by

$$\mathbf{p}_i(t+1) = \begin{cases} \mathbf{p}_i(t) + \mathbf{a}_i(t), & \phi_i(t) = 1 \\ \mathbf{p}_i(t), & \text{otherwise}, \end{cases} \tag{5.2}$$

keeping the UAV stationary if inactive. The evolution of the operational status $\phi_i(t)$ of each UAV is given by

$$\phi_i(t+1) = \begin{cases} 0, & \mathbf{a}_i(t) = [0, 0, -h]^{\mathrm{T}} \\ & \vee\ \phi_i(t) = 0 \\ 1, & \text{otherwise}, \end{cases} \tag{5.3}$$

where the operational status becomes inactive when the UAV has safely landed. The end of the data harvesting mission T is defined as the time slot when all UAVs have reached their terminal state and are not actively operating anymore, i.e. the operational state is $\phi_i(t) = 0$ for all UAVs.

The i-th UAV's battery content evolves according to

$$b_i(t+1) = \begin{cases} b_i(t) - 1, & \phi_i(t) = 1 \\ b_i(t), & \text{otherwise,} \end{cases} \tag{5.4}$$

assuming a constant energy consumption while the UAV is operating and zero energy consumption when operation has terminated. This is a simplification justified by the fact that power consumption for small quadcopter UAVs is dominated by the hovering component. Using the model from [81], the ratio between the additional power necessary for horizontal flight at 10m/s and just hovering could be roughly estimated as 30W/310W $\approx$ 10%, which is negligible. Considering power consumption of on-board computation and communication hardware which does not differ between flight and hovering, the overall difference becomes even smaller. In the following, we will refer to the battery content as remaining flying time, as it is directly equivalent.

The overall multi-UAV mobility model is restricted by the following constraints:

$$\mathbf{p}_i(t) \neq \mathbf{p}_j(t) \vee \phi_j(t) = 0, \quad \forall i, j \in \mathcal{I}, i \neq j, \forall t \tag{5.5a}$$

$$\mathbf{p}_i(t) \notin \mathcal{Z}, \quad \forall i \in \mathcal{I}, \forall t \tag{5.5b}$$

$$b_i(t) \geq 0, \quad \forall i \in \mathcal{I}, \forall t \tag{5.5c}$$

$$\mathbf{p}_i(0) \in \mathcal{L} \wedge z_i(0) = h, \quad \forall i \in \mathcal{I} \tag{5.5d}$$

$$\phi_i(0) = 1, \quad \forall i \in \mathcal{I} \tag{5.5e}$$

The constraint (5.5a) describes collision avoidance among active UAVs with the exception that UAVs can land at the same location. (5.5b) forces the UAVs to avoid collisions with tall obstacles and prevents them from entering NFZs. The constraint (5.5c) limits operation time of the drones, forcing UAVs to end their mission before their battery has run out. Since operation can only be concluded with the landing action as described in (5.3) and the landing action is only available in the landing zone as defined in (5.1), the constraint (5.5c) ensures that each UAV safely lands in the landing zone before their batteries are empty. The starting constraint (5.5d) defines that the UAV start positions are in the start/landing zones and that their starting altitude is h, while (5.5e) ensures that the UAVs start in the active operational state.

5.2.2 Communication Channel Model

Link Performance Model

The link performance model is identical to the one presented in section 4.2.1. Each mission time slot $t \in [0, T]$ is partitioned into a number of $\lambda \in \mathbb{N}$ communication time slots. The communication time index is then $n \in [0, N]$ with $N = \lambda T$, where one communication time slot n is of length $\delta_n = \delta_t / \lambda$ seconds.

The k-th IoT device is located on ground level at $\mathbf{u}_k = [x_k, y_k, 0]^{\mathrm{T}} \in \mathbb{R}^3$ with $k \in \mathcal{K}$ where $|\mathcal{K}| = K$. Each IoT sensor has a finite amount of data $D_k(t) \in \mathbb{R}^+$ that needs to be picked up over the whole mission time $t \in [0, T]$. The device data volume is set to an initial value at the start of the mission $D_k(t = 0) = D_{k,init}$. The data volume of each IoT node evolves depending on the communication time index n over the whole mission time, given by $D_k(n)$ with $n \in [0, N], N = \lambda T$.

We follow the same UAV-to-ground channel model as described in section 2.1.2 and only recall it here with the extension for the multi-UAV case. The maximum achievable information rate at time n for the k-th device is given by

$$R_{i,k}^{\max}(n) = \log_2 \left(1 + \mathrm{SNR}_{i,k}(n)\right). \tag{5.6}$$

Considering the amount of data available at the k-th device $D_k(n)$, the effective information rate is given as

$$R_{i,k}(n) = \begin{cases} R_{i,k}^{\max}(n), & D_k(n) \geq \delta_n R_{i,k}^{\max}(n) \\ D_k(n)/\delta_n, & \text{otherwise.} \end{cases} \tag{5.7}$$

The SNR with transmit power $P_{i,k}$, white Gaussian noise power at the receiver σ^2, UAV-device distance $d_{i,k}$, path loss exponent α_e and $\eta_e \sim \mathcal{N}(0, \sigma_e^2)$ modeled as a Gaussian random variable, is defined as

$$\mathrm{SNR}_{i,k}(n) = \frac{P_{i,k}}{\sigma^2} \cdot d_{i,k}(n)^{-\alpha_e} \cdot 10^{\eta_e/10}. \tag{5.8}$$

Note that the urban environment with the set of obstacles $\mathcal{B}$ hindering free propagation causes a strong dependence of the propagation parameters on the $e \in \{\text{LoS, NLoS}\}$ condition and that (5.8) is the SNR averaged over small scale fading. It is also important to recall that the presented DQN-based trajectory planning approach is model-free and does therefore not rely on any specific channel model. While a more accurate and complex model could be directly used with our approach, the most important features for data collection missions of the urban channel, the dependence of SNR on $d_{i,k}$ and the $e \in \{\text{LoS, NLoS}\}$ condition, are already captured in (5.8).

Multiple Access Protocol

The multiple access protocol is assumed to follow the standard time-division multiple access (TDMA) model when it comes to the communication between one single UAV and the various ground nodes. We further assume that the communication channel between the ground nodes and a given UAV operates on resource blocks (time-frequency slots) that are orthogonal to the channels linking the ground nodes and other UAVs, so that no inter-UAV interference exists in our model and inter-UAV synchronization is not necessary. Hence, the UAVs are similar to base stations that would be assigned orthogonal spectral resources. We also assume that IoT devices are operating in multi-band mode, hence are capable of simultaneously communicating with all UAVs on the set of all orthogonal frequencies. As a consequence, scheduling decisions are not part of the action space. The

number of available orthogonal subchannels for UAV-to-ground communication is one of the variable scenario parameters and equivalent to the number of deployed UAVs.

Designing multiple access protocols for UAV networks is in itself a challenging research problem [44] due to high mobility of the nodes and fast changing link performance and is out of scope for this work. However, our proposed algorithm can in principle be integrated with existing solutions and does not rely on any specific channel model or multiple access protocol. While our model avoids and does not consider inter-UAV interference, we would like to point out that the behavior of the UAV agents that emerges naturally during the learning process of dividing the data collection task geographically, as illustrated in section 5.6.4, would mitigate the influence of interference on the trajectory planning decisions to some extent.

Our scheduling protocol is assumed to follow the max-rate rule: in each communication time slot $n \in [0, N]$, the sensor node $k \in [1, K]$ with the highest $\mathrm{SNR}_{i,k}(n)$ with remaining data to be uploaded is picked by the scheduling algorithm. The TDMA constraint for the scheduling variable $q_{i,k}(n) \in \{0, 1\}$ is given by

$$\sum_{k=1}^{K} q_{i,k}(n) \leq 1, \; n \in [0, N], \forall i \in \mathcal{I}. \tag{5.9}$$

It follows that the k-th device's data volume evolves within one communication time slot according to

$$D_k(n+1) = D_k(n) - \sum_{i=1}^{I} q_{i,k}(n) R_{i,k}(n) \delta_n. \tag{5.10}$$

The achievable throughput for the i-th UAV for one mission time slot $t \in [0, T]$, comprised of λ communication time slots, is the sum of rates achieved in the communication time slots $n \in [\lambda t, \lambda(t+1) - 1]$ over K sensor nodes. It depends on the UAV's operational status $\phi_i(t)$ and is given by

$$C_i(t) = \phi_i(t) \sum_{n=\lambda t}^{\lambda(t+1)-1} \sum_{k=1}^{K} q_{i,k}(n) R_{i,k}(n) \delta_n. \tag{5.11}$$

5.2.3 Optimization Problem

Using the described UAV model in 5.2.1 and communication model in 5.2.2, the central goal of the multi-UAV path planning problem is the maximization of throughput over the whole mission time and over all I deployed UAVs while adhering to mobility constraints (5.5a)-(5.5e) and the scheduling constraint (5.9). The maximization problem is given by

$$\max_{\times_i \mathbf{a}_i(t)} \sum_{t=0}^{T} \sum_{i=1}^{I} C_i(t). \tag{5.12}$$
$$\text{s.t.} \quad (5.5a), (5.5b), (5.5c), (5.5d), (5.5e), (5.9)$$

optimizing over joint actions $\times_i \mathbf{a}_i(t)$.

5.3 Decentralized Partially Observable Markov Decision Process (Dec-POMDP)

To address the aforementioned optimization problem, the simple MDP framework is not applicable anymore, as the multi-agent formulation requires decentralization and the map-processing (as detailed in section 5.4) causes the problem to be only partially observable. Hence, we translate the optimization problem (5.12) to a decentralized partially observable Markov decision process (Dec-POMDP) [117], which is defined through the tuple $(\mathcal{S}, \mathcal{A}_\times, P, R, \Omega_\times, \mathcal{O}, \gamma)$. In the Dec-POMDP, $\mathcal{S}$ describes the state space, $\mathcal{A}_\times = \mathcal{A}^I$ the joint action space, and $P : \mathcal{S} \times \mathcal{A}_\times \times \mathcal{S} \mapsto \mathbb{R}$ the transition probability function. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is the reward function mapping state, individual action, and next state to a real valued reward. The joint observation space is defined through $\Omega_\times = \Omega^I$ and $\mathcal{O} : \mathcal{S} \times \mathcal{I} \mapsto \Omega$ is the observation function mapping state and agents to one agent's individual observation. The discount factor $\gamma \in [0, 1]$ controls the importance of long vs. short term rewards.

5.3.1 State Space

The state space of the multi-agent data collection problem consists of the environment information, the state of the agents, and the state of the devices. It is given as

$$\begin{aligned} \mathcal{S} = \underbrace{\mathcal{L}}_{\substack{\text{Landing}\\ \text{Zones}}} \times \underbrace{\mathcal{Z}}_{\text{NFZs}} \times \underbrace{\mathcal{B}}_{\text{Obstacles}} &\Big\} \text{Environment} \\ \times \underbrace{\mathbb{R}^{I\times 3}}_{\substack{\text{UAV}\\ \text{Positions}}} \times \underbrace{\mathbb{N}^I}_{\substack{\text{Flying}\\ \text{Times}}} \times \underbrace{\mathbb{B}^I}_{\substack{\text{Operational}\\ \text{Status}}} &\Big\} \text{Agents} \\ \times \underbrace{\mathbb{R}^{K\times 3}}_{\substack{\text{Device}\\ \text{Positions}}} \times \underbrace{\mathbb{R}^K}_{\substack{\text{Device}\\ \text{Data}}} &\Big\} \text{Devices} \end{aligned} \tag{5.13}$$

in which the elements $s(t) \in \mathcal{S}$ are

$$s(t) = (\mathbf{M}, \{\mathbf{p}_i(t)\}, \{b_i(t)\}, \{\phi_i(t)\}, \{\mathbf{u}_k\}, \{D_k(t)\}), \tag{5.14}$$

$\forall i \in \mathcal{I}$ and $\forall k \in \mathcal{K}$, in which $\mathbf{M} \in \mathbb{B}^{M\times M\times 3}$ is the tensor representation of the set of start/landing zones $\mathcal{L}$, obstacles and NFZs $\mathcal{Z}$, and obstacles only $\mathcal{B}$. The other elements of the tuple define positions, remaining flying times, and operational status of all agents, as well as positions and available data volume of all IoT devices.

5.3.2 Safety Controller

To enforce the collision avoidance constraint (5.5a) and the NFZ and obstacle avoidance constraint (5.5b), a safety controller is introduced into the system. Additionally, the safety controller enforces the limited action space excluding the *landing* action when the respective agent is not in the landing zone as defined in (5.1). The safety controller

evaluates the action $\mathbf{a}_i(t)$ of agent i and determines if it should be accepted or rejected. If rejected, the resulting safe action is the *hovering* action. The safe action $\mathbf{a}_{s,i}(t)$ is thus defined as

$$\mathbf{a}_{s,i}(t) = \begin{cases} [0,0,0]^{\mathrm{T}}, & \mathbf{p}_i(t) + \mathbf{a}_i(t) \in \mathcal{Z} \\ & \vee\ \mathbf{p}_i(t) + \mathbf{a}_i(t) = \mathbf{p}_j(t) \wedge \phi_j(t) = 1, \\ & \quad \forall j, j \neq i \\ & \vee\ \mathbf{a}_i(t) = [0,0,-h]^{\mathrm{T}}\ \wedge\ \mathbf{p}_i(t) \notin \mathcal{L} \\ \mathbf{a}_i(t), & \text{otherwise.} \end{cases} \tag{5.15}$$

Without path planning capabilities, the safety controller cannot enforce the flying time and safe landing constraint in (5.5c). Therefore, we relax the hard constraint on flight time by adding a high penalty on not landing in time instead. In the simulation, a crashed agent, i.e. an agent with $b_i(t) < 0$, is defined as not operational.

5.3.3 Reward Function

The reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ of the Dec-POMDP is comprised of the following elements:

$$r_i(t) = \alpha \sum_{k \in \mathcal{K}} \Big(D_k(t+1) - D_k(t) \Big) + \beta_i(t) + \gamma_i(t) + \epsilon. \tag{5.16}$$

The first term of the sum is a collective reward for the collected data from all devices by all agents within mission time slot t. It is parameterized through the data collection multiplier α. This is the only part of the reward function that is shared among all agents. The second addend is an individual penalty when the safety controller rejects an action and given through

$$\beta_i(t) = \begin{cases} \beta, & \mathbf{a}_i(t) \neq \mathbf{a}_{i,s}(t) \\ 0, & \text{otherwise.} \end{cases} \tag{5.17}$$

It is parameterized through the safety penalty β. The third term is the individual penalty for not landing in time given by

$$\gamma_i(t) = \begin{cases} \gamma, & b_i(t+1) = 0 \wedge \mathbf{p}_i(t+1) = [\cdot, \cdot, h]^{\mathrm{T}} \\ 0, & \text{otherwise.} \end{cases} \tag{5.18}$$

and parameterized through the crashing penalty γ. The last term is a constant movement penalty parameterized through ϵ, which is supposed to incentivize the agents to reduce their flying time and prioritize efficient trajectories.

5.4 Map-Processing and Observation Space

To aid the agents in interpreting the large state space given in (5.13), we implement two map processing steps. The first is centering the map around the agent's position, shown in the previous chapter 4 to significantly improve the agent's learning performance. This benefit is a consequence of neurons in the layer after the convolutional layers (compare

Fig. 5.3) corresponding to features *relative* to the agent's position, rather than to *absolute* positions if the map is not centered. This is advantageous as one agent's actions are solely based on its relative position to features, e.g. its distance to sensor devices. The downside of this approach is that it increases the size of the maps and the observation space even further, therefore requiring larger networks with more trainable parameters.

The second map processing step is to present the centered map as a compressed global and uncompressed but cropped local map as introduced in [79]. In path planning, as distant features lead to general direction decisions while close features lead to immediate actions such as collision avoidance, the level of detail passed to the agent for distant objects can be less than for close objects. The advantage is that the compression of the global map reduces the necessary neural network size considerably.

This reduction in network size directly translates to a reduction in computational load. Table 5.1 shows the number of floating point operations needed for each of the two maps under different map processing regimes as given by the TensorFlow graph profiler. Only centering increases the computational load considerably, while global-local map processing offsets the increase and reduces floating point operations considerably. Considering that modern embedded processors operate in the region of giga floating point operations, it seems realistic that the required processing can be carried out even on small and energy-limited UAVs. The mathematical descriptions of the map processing functions and the observation space are detailed in the following.

Map	No Processing	Centering	Centering + Global-Local
Manhattan32	15	80	7.7
Urban50	45	217	6.5

Table 5.1 – Million floating point operations (MFLOPs) needed for inference of the networks based on map-processing.

5.4.1 Map-Processing

For ease of exposition, we introduce the 2D projections of the UAV and IoT device positions on the ground, $\tilde{\mathbf{u}}_k \in \mathbb{N}^2$ and $\tilde{\mathbf{p}}_k \in \mathbb{N}^2$ respectively, given by

$$\tilde{\mathbf{u}}_k = \left\lfloor \begin{pmatrix} \frac{1}{c} & 0 & 0 \\ 0 & \frac{1}{c} & 0 \end{pmatrix} \mathbf{u}_k \right\rceil, \quad \tilde{\mathbf{p}}_i = \left\lfloor \begin{pmatrix} \frac{1}{c} & 0 & 0 \\ 0 & \frac{1}{c} & 0 \end{pmatrix} \mathbf{p}_i \right\rceil \tag{5.19}$$

rounded to integer grid coordinates.

Mapping

The centering and global-local mapping algorithms are based on map-layer representations of the state space. To represent any state with a spatial aspect given by a position and a corresponding value as a map-layer, we define a general mapping function

$$f_{\text{mapping}} : \mathbb{N}^{Q\times 2} \times \mathbb{R}^Q \mapsto \mathbb{R}^{M\times M}. \tag{5.20}$$

In this function, a map layer $\mathbf{A} \in \mathbb{R}^{M \times M}$ is defined as

$$\mathbf{A} = f_{\text{mapping}}(\{\tilde{\mathbf{p}}_q\}, \{v_q\}), \tag{5.21}$$

with a set of grid coordinates $\{\tilde{\mathbf{p}}_q\}$ and a set of corresponding values $\{v_q\}$. The elements of $\mathbf{A}$ are given through

$$a_{\tilde{p}_{q,0}, \tilde{p}_{q,1}} = v_q, \quad \forall q \in [0, ..., Q-1] \tag{5.22}$$

or 0 if the index is not in the grid coordinates. With this general function, we define the map-layers

$$\mathbf{D}(t) = f_{\text{mapping}}(\{\tilde{\mathbf{u}}_k\}, \{D_k(t)\}) \tag{5.23a}$$
$$\mathbf{B}(t) = f_{\text{mapping}}(\{\tilde{\mathbf{p}}_i(t)\}, \{b_i(t)\}) \tag{5.23b}$$
$$\Phi(t) = f_{\text{mapping}}(\{\tilde{\mathbf{p}}_i(t)\}, \{\phi_i(t)\}) \tag{5.23c}$$

for device data, UAV flying times, and UAV operational status respectively. If the map-layers are of same type they can be stacked to form a tensor of $\mathbb{R}^{M \times M \times n}$ for ease of representation.

Map Centering

Given a tensor $\mathbf{A} \in \mathbb{R}^{M \times M \times n}$ describing the map-layers, a centered tensor $\mathbf{B} \in \mathbb{R}^{M_c \times M_c \times n}$ with $M_c = 2M - 1$ is defined through

$$\mathbf{B} = f_{\text{center}}(\mathbf{A}, \tilde{\mathbf{p}}, \mathbf{x}_{\text{pad}}), \tag{5.24}$$

with the centering function defined as

$$f_{\text{center}} : \mathbb{R}^{M \times M \times n} \times \mathbb{N}^2 \times \mathbb{R}^n \mapsto \mathbb{R}^{M_c \times M_c \times n}. \tag{5.25}$$

The elements of $\mathbf{B}$ with respect to the elements of $\mathbf{A}$ are defined as

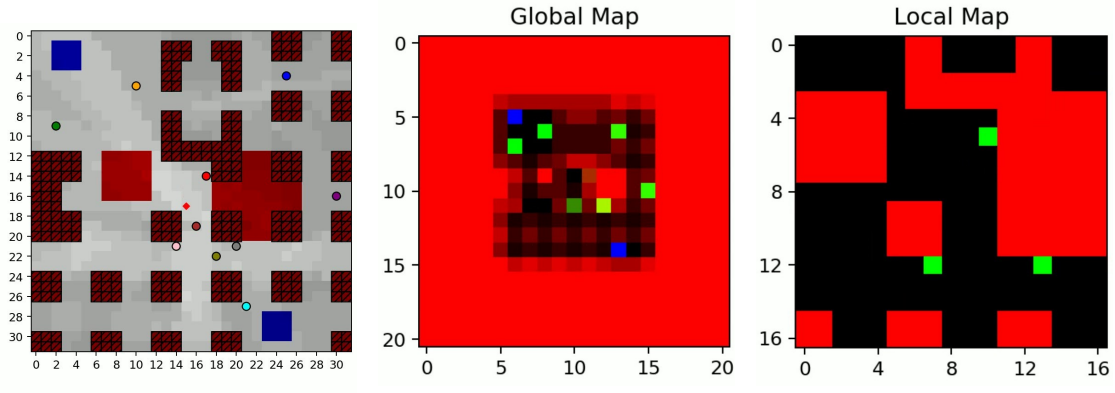
$$\mathbf{b}_{i,j} = \begin{cases} \mathbf{a}_{i+\tilde{p}_0-M+1, j+\tilde{p}_1-M+1}, & M \leq i + \tilde{p}_0 + 1 < 2M \\ & \wedge\ M \leq j + \tilde{p}_1 + 1 < 2M \\ \mathbf{x}_{\text{pad}}, & \text{otherwise}, \end{cases} \tag{5.26}$$

effectively padding the map layers of $\mathbf{A}$ with the padding value $\mathbf{x}_{\text{pad}}$. Note that $\mathbf{a}_{i,j}$, $\mathbf{b}_{i,j}$, and $\mathbf{x}_{\text{pad}}$ are vector valued of dimension $\mathbb{R}^n$. An illustration of the centering on a 16×16 map ($M = 16$, $M_c = 31$) was given in the previous chapter and can be seen in Figure 4.3 with the legend in Tab. 4.2.

Global-Local Map

The tensor $\mathbf{B} \in \mathbb{R}^{M_c \times M_c \times n}$ resulting from the map centering function is processed in two ways. The first is creating a local map according to

$$\mathbf{X} = f_{\text{local}}(\mathbf{B}, l) \tag{5.27}$$



(a) Original complete map.

(b) Compressed and centered global map.

(c) Uncompressed but cropped local map.

Figure 5.1 – Visualization of the global-local map processing. The active agent's position is marked by a red spot on the original map 5.1a.

with the local map function defined by

$$f_{\text{local}} : \mathbb{R}^{M_c \times M_c \times n} \times \mathbb{N} \mapsto \mathbb{R}^{l \times l \times n}. \tag{5.28}$$

The elements of $\mathbf{X}$ with respect to the elements of $\mathbf{B}$ are defined as

$$\mathbf{x}_{i,j} = \mathbf{b}_{i+M-\lceil \frac{l}{2} \rceil, j+M-\lceil \frac{l}{2} \rceil} \tag{5.29}$$

This operation is effectively a central crop of size $l \times l$.

The second processing creates a global map according to

$$\mathbf{Y} = f_{\text{global}}(\mathbf{B}, g) \tag{5.30}$$

with the global map function

$$f_{\text{global}} : \mathbb{R}^{M_c \times M_c \times n} \times \mathbb{N} \mapsto \mathbb{R}^{\lfloor \frac{M_c}{g} \rfloor \times \lfloor \frac{M_c}{g} \rfloor \times n} \tag{5.31}$$

The elements of $\mathbf{Y}$ with respect to the elements of $\mathbf{B}$ are defined as

$$\mathbf{y}_{i,j} = \frac{1}{g^2} \sum_{u=0}^{g-1} \sum_{v=0}^{g-1} \mathbf{b}_{gi+u, gj+v} \tag{5.32}$$

This operation is equal to an average pooling operation with pooling cell size g.

The functions f_{local} and f_{global} are parameterized through l and g, respectively. Increasing l increases the size of the local map, whereas increasing g increases the size of the average pooling cells, therefore decreasing the size of the global map. The global-local map processing is visualized in Fig. 5.1.

5.4.2 Observation Space

Using the map processing functions, the observation space can be defined. The observation space Ω, which is the input space to the agent, is given as

$$\Omega = \underbrace{\Omega_l}_{\substack{\text{Local}\\\text{Map}}} \times \underbrace{\Omega_g}_{\substack{\text{Global}\\\text{Map}}} \times \underbrace{\mathbb{N}}_{\substack{\text{Flying}\\\text{Time}}}$$

containing the local map

$$\Omega_l = \mathbb{B}^{l\times l\times 3} \times \mathbb{R}^{l\times l} \times \mathbb{N}^{l\times l} \times \mathbb{B}^{l\times l}$$

and the global map

$$\Omega_g = \mathbb{R}^{\bar{g}\times\bar{g}\times 3} \times \mathbb{R}^{\bar{g}\times\bar{g}} \times \mathbb{R}^{\bar{g}\times\bar{g}} \times \mathbb{R}^{\bar{g}\times\bar{g}}.$$

with $\bar{g} = \lfloor \frac{M_c}{g} \rfloor$. Note that the compression of the global map through average pooling transforms all map layers into $\mathbb{R}$. Observations $o_i(t) \in \Omega$ are defined through the tuple

$$\begin{aligned} o_i(t) = (&\mathbf{M}_{l,i}(t), \mathbf{D}_{l,i}(t), \mathbf{B}_{l,i}(t), \Phi_{l,i}(t),\\ &\mathbf{M}_{g,i}(t), \mathbf{D}_{g,i}(t), \mathbf{B}_{g,i}(t), \Phi_{g,i}(t), b_i(t)). \end{aligned} \tag{5.33}$$

In one observation tuple, $\mathbf{M}_{l,i}(t)$ is the local observation of agent i of the environment, $\mathbf{D}_{l,i}(t)$ is the local observation of the data to be collected, $\mathbf{B}_{l,i}(t)$ is the local observation of the remaining flying time of all agents, and $\Phi_{l,i}(t)$ is the local observation of the operational status of the agents. $\mathbf{M}_{g,i}(t)$, $\mathbf{D}_{g,i}(t)$, $\mathbf{B}_{g,i}(t)$, and $\Phi_{g,i}(t)$ are the respective global observations. $b_i(t)$ is the remaining flying time of agent i, which is equal to the one in the state space. Note that the environment map's local and global observations are dependent on time, as they are centered around the UAV's time-dependent position. Additionally, it should be noted that the remaining flying time of agent i is given in the center of $\mathbf{B}_{l,i}(t)$ and additionally as a scalar $b_i(t)$. This redundancy in representation helps the agent to interpret the remaining flying time.

Consequently, the complete mapping from state to observation space is given by

$$\mathcal{O} : \mathcal{S} \times \mathcal{I} \mapsto \Omega \tag{5.34}$$

in which the elements of $o_i(t)$ are defined as follows:

$$\mathbf{M}_{l,i}(t) = f_{\text{local}}(f_{\text{center}}(\mathbf{M}, \mathbf{p}_i(t), [0,1,1]^{\mathrm{T}}), l) \tag{5.35a}$$
$$\mathbf{D}_{l,i}(t) = f_{\text{local}}(f_{\text{center}}(\mathbf{D}(t), \mathbf{p}_i(t), 0), l) \tag{5.35b}$$
$$\mathbf{B}_{l,i}(t) = f_{\text{local}}(f_{\text{center}}(\mathbf{B}(t), \mathbf{p}_i(t), 0), l) \tag{5.35c}$$
$$\Phi_{l,i}(t) = f_{\text{local}}(f_{\text{center}}(\Phi(t), \mathbf{p}_i(t), 0), l) \tag{5.35d}$$
$$\mathbf{M}_{g,i}(t) = f_{\text{global}}(f_{\text{center}}(\mathbf{M}, \mathbf{p}_i(t), [0,1,1]^{\mathrm{T}}), g) \tag{5.35e}$$
$$\mathbf{D}_{g,i}(t) = f_{\text{global}}(f_{\text{center}}(\mathbf{D}(t), \mathbf{p}_i(t), 0), g) \tag{5.35f}$$
$$\mathbf{B}_{g,i}(t) = f_{\text{global}}(f_{\text{center}}(\mathbf{B}(t), \mathbf{p}_i(t), 0), g) \tag{5.35g}$$
$$\Phi_{g,i}(t) = f_{\text{global}}(f_{\text{center}}(\Phi(t), \mathbf{p}_i(t), 0), g) \tag{5.35h}$$

By passing the observation space Ω into the agent instead of the state space $\mathcal{S}$ as done in the previous chapter 4, the presented path planning problem is artificially converted into a partially observable MDP. Partial observability is a consequence of the restricted size of the local map and the compression of the global map. However, as shown in [79], partial observability does not render the problem infeasible, even for a memory-less agent. Instead, the compression greatly reduces the neural network's size, leading to a significant reduction in training time.

5.5 Multi-Agent Reinforcement Learning

5.5.1 Multi-Agent Q-learning

In principle, we apply the DDQN algorithm as introduced in section 4.4 directly to the presented multi-UAV problem. Loss function and target are given by equations (4.4) and (4.5) respectively. The following provides a characterization of the proposed DDQN multi-agent algorithm.

The original table-based Q-learning algorithm was extended to the cooperative multi-agent setting by Claus and Boutilier in 1998 [118]. Without changing the underlying principle, it can also be applied to DDQN-based multi-agent cooperation. With the taxonomy from [119], our agents can be classified as homogeneous and non-communicating. Homogeneity is a consequence of deploying a team of identical UAVs with the same internal structure, domain knowledge, and identical action spaces. Non-communication is to be interpreted in a multi-agent system sense, i.e. that the agents can not coordinate their actions or choose what to communicate. However, as they all perceive state information that includes other UAVs' positions, in a practical sense, position information would most likely be communicated via the command and control links of the UAVs, that especially autonomous UAVs would have to maintain for regulatory purposes in any case.

The best way to describe our learning approach is by centralized training with decentralized execution (CTDE). As DDQN learning requires an extensive experience database to train the neural networks on, it is reasonable to assume that the experiences made by independently acting agents can be centrally pooled throughout the training phase. After training has concluded, the control systems are individually deployed to the distributed drone agents. The rationale behind this concept is that we investigate a team of homogeneous UAVs with identical capabilities and tasks, therefore all experiences are useful for the training of all agents. In a real-world deployment of a team of quadcopter UAVs, all UAVs would be required to regularly return to a charging station, as flying time remains strongly limited by available on-board battery capacity. While being recharged, the UAVs would upload their experience data to a central server with larger memory and computation resources.

Our setting can not be characterized as fully cooperative as our agents do not share a common reward [120]. Instead, each agent has an individual but identical reward function. As the main component of the reward function is based on the jointly collected data from the IoT devices described in Section 5.3.3, they do share a common goal, leading to the classification of our setting as a simple cooperative or individualistic setting.

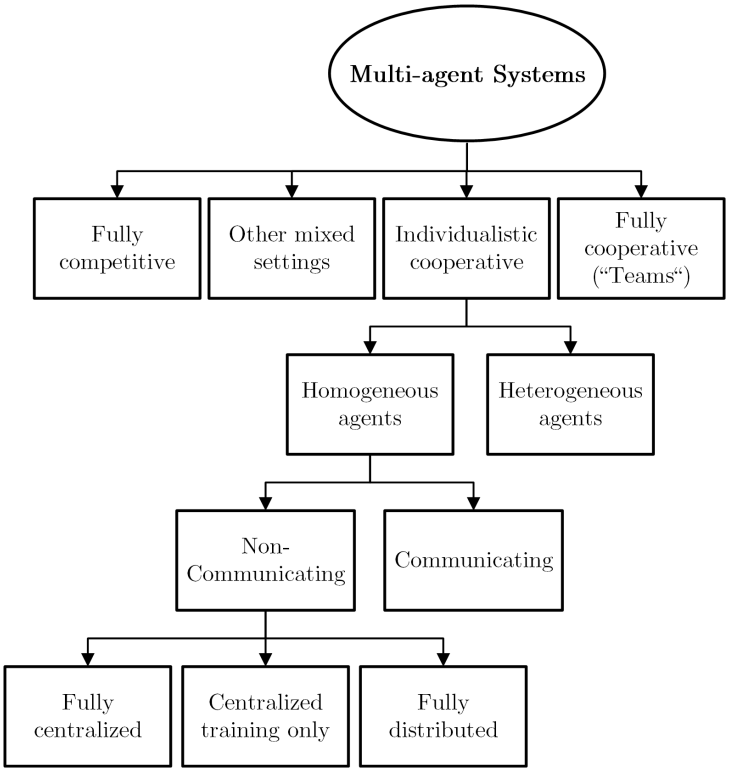


Figure 5.2 – Classification of the presented approach in the context of a taxonomy of MARL methods.

Fig. 5.2 gives an overview of the taxonomy of MARL approaches.

5.5.2 Neural Network Model

We use a neural network model similar to the one presented in the previous chapter and depicted in Fig. 4.4 . Fig. 5.3 shows the DDQN structure and the map centering and global-local map processing. The map information of the environment, NFZs, obstacles, and start/landing area is stacked with the IoT device map and the map with the other UAVs' flying times and operational status. According to Section 5.4.1, the map is centered on the UAV's position and split into a global and local map. The global and local maps are fed through convolutional layers with ReLU activation and then flattened and concatenated with the scalar input indicating battery content or remaining flight time. After passing through fully connected layers with ReLU activation, the data reaches the last fully-connected layer of size $|\mathcal{A}|$ without activation function, directly representing the Q-values for each action given the input observation. The greedy policy is used for evaluation and the soft-max policy for training, given by equations (2.15) and (2.16) respectively. Hyperparameters are listed in Tab. 5.2.

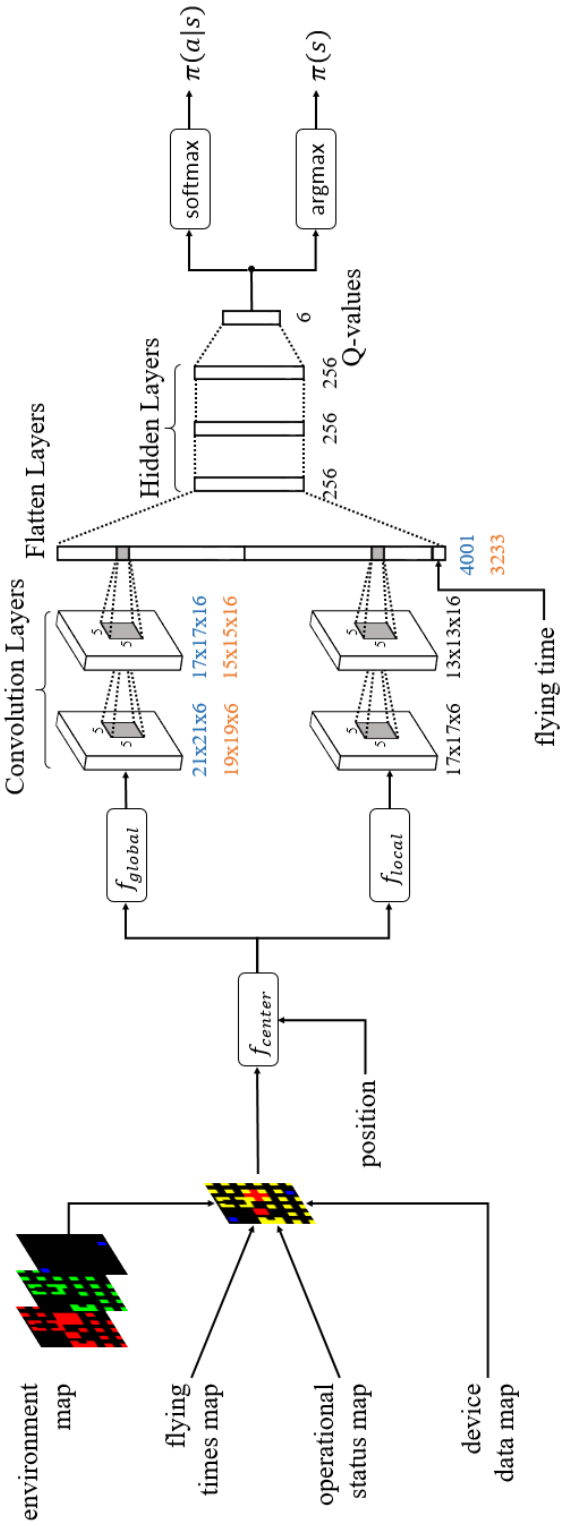


Figure 5.3 – DDQN architecture with map centering and global and local map processing. Layer sizes are shown in in blue for the smaller 'Manhattan32' scenario and orange for the larger 'Urban50' scenario.

Parameter	32×32	50×50	Description
$\|\theta\|$	1,175,302	978,694	trainable parameters
$N_{\max}$	3,000,000	4,000,000	maximum training steps
l	17	17	local map scaling
g	3	5	global map scaling
n_c	2		number of conv. layers
n_k	16		number of kernels
s_k	5		conv. kernel width
$\|\mathcal{D}\|$	50,000		replay memory buffer size
m	128		minibatch size
τ	0.005		soft update factor in (2.14)
γ	0.95		discount factor in (4.5)
β	0.1		temperature parameter (2.16)

Table 5.2 – DDQN Hyperparameters for 32×32 and 50×50 maps.

5.6 Simulations

5.6.1 Simulation Setup

In line with the previous chapter 4, we aim to provide an algorithm[1] that is able to generalize the learned UAV control policy over a the parameter space that defines a data collection mission. In addition to the parameters listed in section 4.6.1, i.e. number and position of IoT sensor nodes, data amount, flying time, and starting positions, we also generalize over the number of deployed UAVs.

The exact value ranges from which these parameters are sampled are given in the following Sections 5.6.3 and 5.6.4 depending on the map. We deploy our system on two different maps. In 'Manhattan32', the UAVs fly inside 'urban canyons' through a dense city environment discretized into 32×32 cells, whereas 'Urban50' is an example of a less dense but larger 50×50 urban area. Note that we only trained a single agent on each of these maps, meaning that all results discussed in the following are a result of only two trained agents. Generalization over this large parameter space is possible in part due to the learning efficiency benefits from feeding map information centered on the agents' respective positions into the network, as we have described previously in chapter 4. The evaluation strategy is identical to the one described before in section 4.6.1.

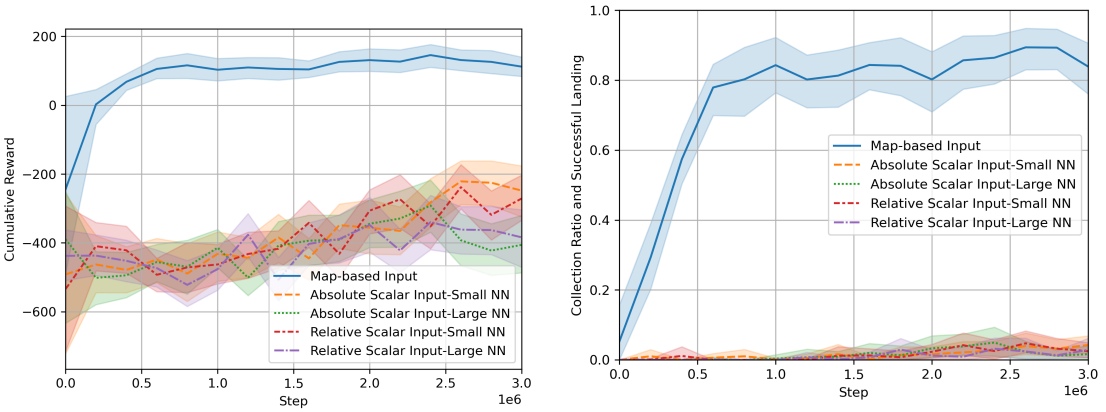
Irrespective of the map, the grid cell size is $c = 10$m and the UAVs fly at a constant altitude of $h = 10$m over city streets. The UAVs are not allowed to fly over tall buildings, enter NFZs, or leave the respective grid worlds. Each mission time slot $t \in [0, T]$ contains $\lambda = 4$ scheduled communication time slots $n \in [0, N]$. Propagation parameters (see

[1]The Python code for this work is available under `https://github.com/hbayerlein/uav_data_harvesting`.

2.1.2) are again chosen in line with [110] according to the urban micro scenario with $\alpha_{\text{LoS}} = 2.27$, $\alpha_{\text{NLoS}} = 3.64$, $\sigma^2_{\text{LoS}} = 2$ and $\sigma^2_{\text{NLoS}} = 5$.

Due to the drones flying below or slightly above building height, the wireless channel is characterized by strong LoS/NLoS dependency and shadowing. The shadowing maps used for simulation of the environment were computed using ray tracing from and to the center points of cells based on a variation of Bresenham's line algorithm. Transmission and noise powers are normalized by defining a cell-edge SNR for each map.

5.6.2 Training with Map-based vs. Scalar Inputs



(a) Cumulative reward per episode

(b) Data collection ratio with successful landing per episode

Figure 5.4 – Training process comparison between *map-based* DRL path planning and *scalar* input DRL path planning. Scalar inputs to the neural networks (NNs) are either encoded as *absolute* coordinate values or *relative* distances from the respective agent. We compare two different scalar input network architectures with *large* and *small* numbers of trainable parameters. The average and 99% quantiles are shown with metrics per training episode grouped in bins of 2e5 step width. Note that the metrics are plotted over training steps as training episode length is variable.

In this section, we show that our map-based approach has a good complexity-performance trade-off in comparison to classical scalar input neural network approaches from the literature despite the added complexity through map-processing. To illustrate that it is in fact imperative for training success to feed map information instead of concatenated scalar values as state input to the agent, we extend our previous analysis from chapter 4 and [79] by comparing our proposed centered global-local map approach to agents trained only on scalar inputs. This is not an entirely fair comparison as the location of NFZs, buildings, and start/landing zones can not be efficiently represented by scalar inputs and must be therefore learned by the scalar agents through trial and error. However, the comparison illustrates the need for state space representations that are different from the traditional scalar inputs and confirms that scalar agents are not

able to solve the multi-UAV path planning problem over the large scenario parameter space presented. Conversely, the alternative comparison of map-based and scalar agents trained on a *single* data harvesting scenario would not yield meaningful results as our method is specifically designed to generalize over a large variety of scenarios and would require tweaking in exploration behavior and reward balance to find the optimal solution to a single scenario. Note that most of the previous work discussed in section 5.1.1 is precisely focused on finding optimal DRL solutions to single scenario instances.

The observation space of the agents trained with concatenated scalar inputs is described by

$$\begin{aligned} \mathcal{O}_{\text{scalar}} = & \underbrace{\mathbb{N}^2}_{\substack{\text{Ego} \\ \text{Position}}} \times \underbrace{\mathbb{N}}_{\substack{\text{Ego Flying} \\ \text{Time}}} && \Big\}\text{Ego agent} \\ & \times \underbrace{\mathbb{N}^{I\times 2}}_{\substack{\text{UAV} \\ \text{Positions}}} \times \underbrace{\mathbb{N}^{I}}_{\substack{\text{Flying} \\ \text{Times}}} \times \underbrace{\mathbb{B}^{I}}_{\substack{\text{Operational} \\ \text{Status}}} && \Big\}\text{Other agents} \\ & \times \underbrace{\mathbb{N}^{K\times 2}}_{\substack{\text{Device} \\ \text{Positions}}} \times \underbrace{\mathbb{R}^{K}}_{\substack{\text{Device} \\ \text{Data}}} && \Big\}\text{Devices} \end{aligned} \tag{5.36}$$

forming the input of the neural network as concatenated scalar values. Since the number of agents and devices is variable, the scalar input size is fixed to the maximum number of agents and devices. The agent and device positions are either represented as *absolute* values in the grid coordinate frame or *relative* as distances from the ego agent. The neural network is either *small*, containing the same number of hidden layers as in Fig. 5.3, or *large*, for which the number and size of hidden layers is adapted such that the network has as many trainable parameters as the map-based 32×32 agent in Tab. 5.2.

Fig. 5.4 shows the cumulative reward and the collection ratio with successful landing metric over training time on the 'Manhattan32' map for the five different network architectures. It is clear that the scalar agents are not able to effectively adapt to the changing scenario conditions. The *small* neural network agents seem to have a slight edge over the *large* agents, but representing the positions as *absolute* or *relative* does not influence the results.

Referring further to Fig. 5.4, the map-based agent converges to final performance metric levels after the first 20% of the training steps. However we observed that additional training is needed after that to optimize the trajectories in a more subtle way for flight time efficiency and multi-UAV coordination. The overall training time for the full 3 million training steps was around 40 hours on a 2017 Nvidia Titan Xp GPU.

5.6.3 'Manhattan32' Scenario

The scenario, as shown in Fig. 5.5 is defined by a Manhattan-like city structure containing mostly regularly distributed city blocks with streets in between, as well as two NFZ districts and an open space in the upper left corner, divided into $M = 32$ cells in each grid direction. This is double the size of the otherwise similarly designed single UAV scenario in chapter 4. We are able to solve the larger scenario without increasing network size,

thanks to the global-local map approach. The value ranges from which the randomized scenario parameters are chosen as follows: number of deployed UAVs $I \in \{1, 2, 3\}$, number of IoT sensors $K \in [3, 10]$, data volume to be collected $D_{k,init} \in [5.0, 20.0]$ data units per device, maximum flying time $b_0 \in [50, 150]$ steps, and 18 possible starting positions. The IoT device positions are randomized throughout the unoccupied map space.

Metric	Manhattan32	Urban50
Successful Landing	99.4%	98.8%
Collection Ratio	88.0%	82.1%
Collection Ratio and Landed	87.5%	81.1%

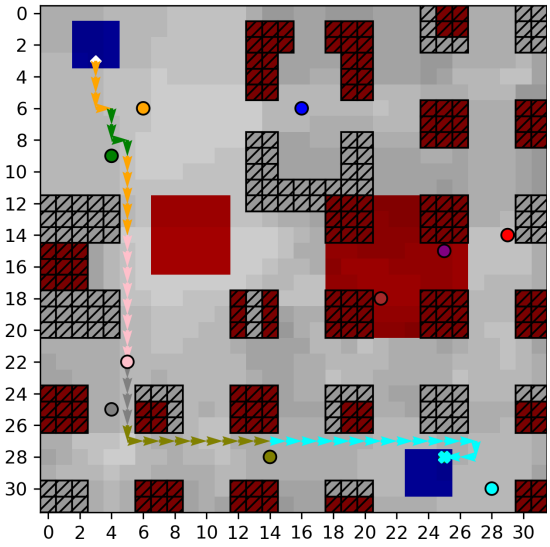
Table 5.3 – Performance metrics averaged over 1000 random scenario Monte Carlo iterations.

The performance on both maps is evaluated using Monte Carlo simulations on their respective full range of scenario parameters with overall average performance metrics shown in Table 5.3. Both agents show a similarly high successful landing performance. It is expected that the collection ratio cannot reach 100% in some scenario instances depending on the randomly assigned maximum flying time, number of deployed UAVs, and IoT device parameters.

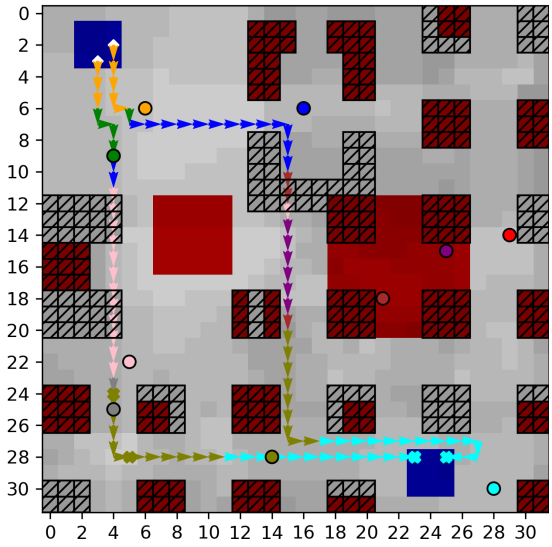
	Symbol	Description
DQN Input		Start and landing zone
		Regulatory no-fly zone (NFZ)
		Tall buildings* (UAVs cannot fly over)
		Small buildings* (UAVs can fly over)
		IoT device
		Other agents
		*all buildings obstruct wireless links
Visualization		Summation of building shadows
		Starting and landing positions during an episode
		UAV movement while comm. with green device
		Hovering while comm. with green device
		Actions without comm. (all data collected)

Table 5.4 – Legend for scenario plots with small and tall buildings.

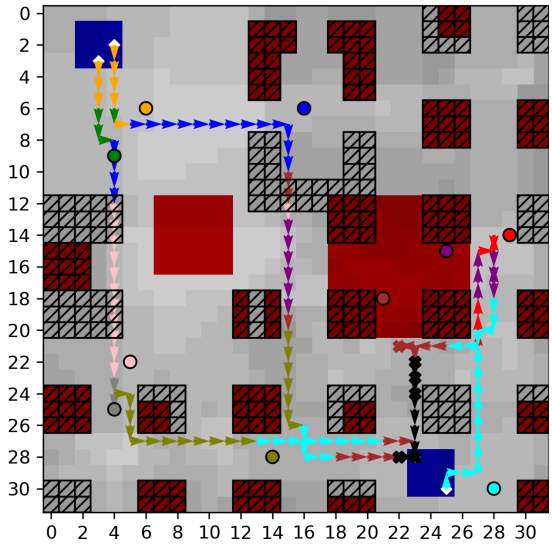
In Fig. 5.5, three scenario instances chosen from the random Monte Carlo evaluation for number of deployed UAVs $I \in \{1, 2, 3\}$ for 5.5a through 5.5c illustrate how the path planning adapts to the increasing number of deployed UAVs. All other scenario parameters are kept fixed. It is a fairly complicated scenario with a large number of IoT



(a) $I = 1$ agent, data collection ratio 56.2%



(b) $I = 2$ agents, data collection ratio 76.5%



(c) $I = 3$ agents, data collection ratio 100%

Figure 5.5 – Example trajectories for 'Manhattan32' map with $K = 10$ IoT devices, all with $D_{k,init} = 15$ data units to be picked up and a maximum flying time of $b_0 = 60$ steps. The color of the UAV movement arrows shows with which device the drone is communicating at the time (see legend in Tab. 5.4).

devices spread out over the whole map, including the brown and purple device inside an NFZ. The agents have no access to the shadowing map and have to deduce shadowing effects from building and device positions.

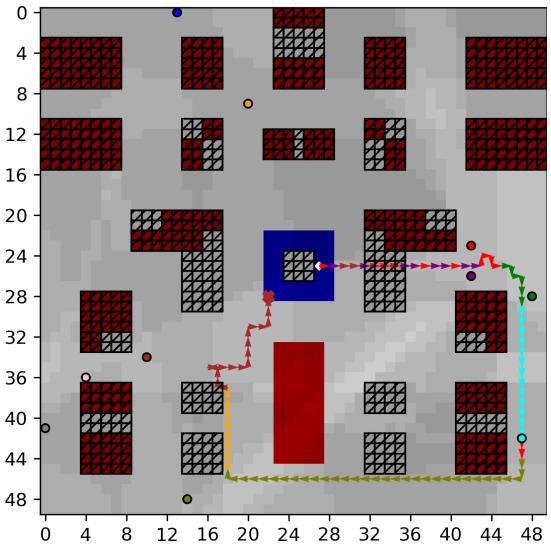
In Fig. 5.5a, only one UAV starting in the upper left corner is deployed. Due to its flight time constraint, the agent ignores the blue, red, purple, and brown IoT devices while collecting all data from the other devices on an efficient trajectory to the landing zone in the lower right corner. When a second UAV is deployed in Fig. 5.5b, the data collection ratio increases to 76.5%. While the first UAV's behavior is almost unchanged compared to the single UAV deployment, the second UAV flies to the landing zone in the lower right corner via an alternative trajectory collecting data from the devices the first UAV ignores. With the number of deployed UAVs increased to three (two starting from the upper left and one from the lower right zone) in Fig. 5.5c, all data can be collected. The second UAV modifies its behavior slightly, accounting for the fact that the third UAV can collect the cyan device's data now. The three UAVs divide the data harvesting task fairly among themselves, leading to full data collection with in-time landing on efficient trajectories while avoiding the NFZs.

5.6.4 'Urban50' Scenario

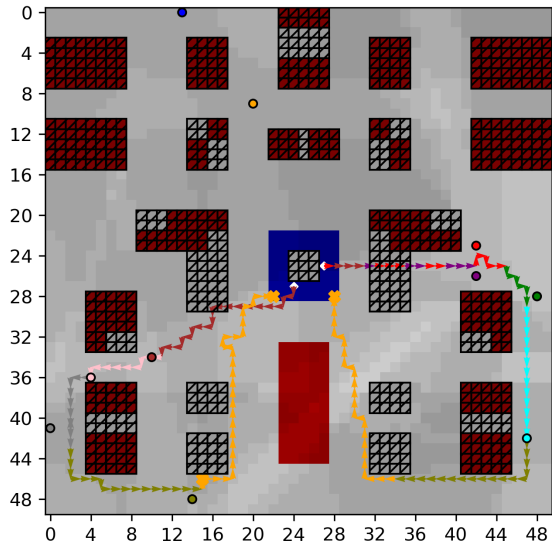
Fig. 5.6 shows three example trajectories for UAV counts of $I \in \{1, 2, 3\}$ for 5.6a through 5.6c in the large 50×50 urban map. The scenario is defined by an urban structure containing irregularly shaped large buildings, city blocks and an NFZ, with the start/landing zone surrounding a building in the center, divided into $M = 50$ cells in each grid direction. The map has an order of magnitude more cells than the scenarios in chapter 4. The ranges for randomized scenario parameters are chosen as follows: number of deployed UAVs $I \in \{1, 2, 3\}$, number of IoT sensors $K \in [5, 10]$, data volume to be collected $D_{k,init} \in [5.0, 20.0]$ data units, maximum flying time $b_0 \in [100, 200]$ steps, and 40 possible starting positions. The IoT device positions are randomized throughout the unoccupied map space.

Fig. 5.6a shows a single agent trying to collect as much data as possible during the allocated maximum flying time. The agent focuses on collecting the data from the relatively easily reachable device clusters on the right and lower half before safely landing. With a second UAV assigned to the mission as shown in Fig. 5.6b, one UAV services the devices on the lower left of the map, while the other one collects data from the devices on the lower right, ignoring the more isolated blue and orange device in the top half of the map. A third UAV makes it possible to divide the map into three sectors and collect all IoT device data, as shown in Fig. 5.6c.

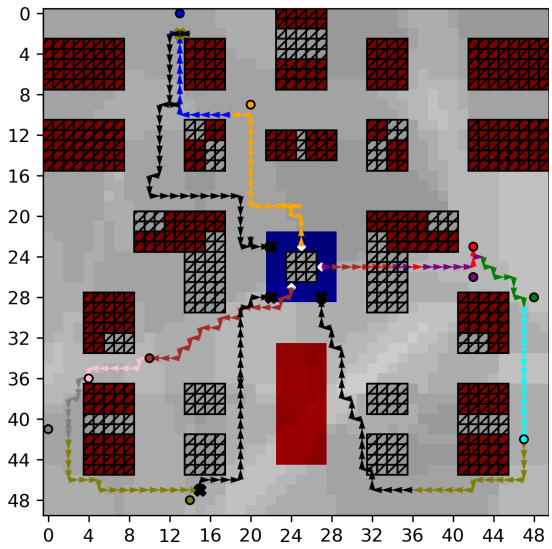
This map's primary purpose is to showcase the significant advantages in terms of training time efficiency and the required network size from the global-local map approach. Thanks to a higher global map scaling or compression factor g (see Table 5.2), the number of trainable parameters of the network employed in this scenario is even smaller compared to the network used for 'Manhattan32'. A network without a map scaling approach would need to contain 34,061,446 trainable parameters, hence a size that is infeasible to train using reasonable resources.



(a) $I = 1$ agent, data collection ratio 41.8%



(b) $I = 2$ agents, data collection ratio 80.2%



(c) $I = 3$ agents, data collection ratio 100.0%

Figure 5.6 – Example trajectories for 'Urban50' map with $K = 10$ IoT devices, all with $D_{k,init} = 15$ data units to be picked up and a maximum flying time of $b_0 = 100$ steps for all UAVs (legend in Tab. 5.4).

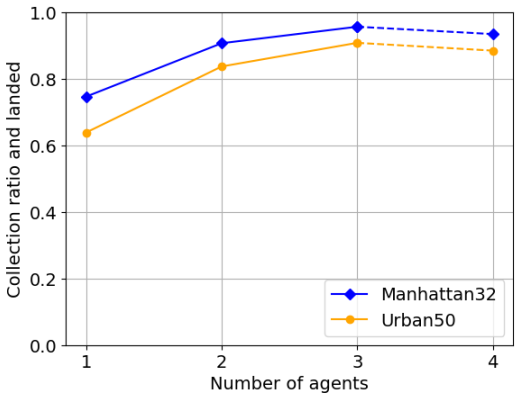
5.6.5 Influence of Scenario Parameters on Performance and System-level Benefits

An advantage of our approach to learn a generalized path planning policy over various scenario parameters is the possibility to analyze how performance indicators change over a variable parameter space. This makes it possible for an operator to decide on system-level trade-offs, e.g. how many drones to deploy vs. collected data volume. An excellent example that we found for the 'Manhattan32' map was that deploying multiple coordinating drones can trade-off the cost of extra equipment (i.e. the extra drones) for substantially reduced mission time. For instance, it takes twice the flying time ($b_0 = 150$) for a single UAV to complete the data collection mission that two coordinating UAVs will require ($b_0 = 75$) to conclude successfully. Specifically, that means that for both scenarios the average data collection ratio with in-time successful landing stays at the same performance level of around 88%.

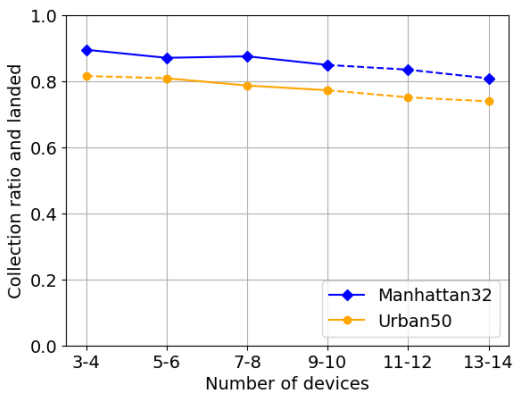
We first analyze the performance of the agent in Fig. 5.7 within the training range of the scenario parameters (solid lines), then extend the analysis to out-of-distribution scenarios (dashed lines) in the last paragraph of this section. Fig. 5.7 shows the influence of single scenario parameters on the average data collection ratio with successful landing of all agents. As already evident from the example trajectories shown previously, Fig. 5.7a indicates the increase in collection performance when more UAVs are deployed. At the same time, more UAVs lead to increased collision avoidance requirements, as we observed through more safety controller activations in the early training phases. As IoT devices are positioned randomly throughout the unoccupied map space, an increase in devices leads to more complex trajectory requirements and a drop in performance, as depicted in Fig. 5.7b.

Fig. 5.7c shows the influence of increasing initial data volume per device on the overall collection performance. It appears that higher initial data volumes per device are beneficial roughly up to the point of $D_{k,init} \in [10, 12.5]$ data units, after which flying time constraints force the UAVs to abandon some of the data, and the collection ratio shows a slightly negative trend. An increase in available flying time is clearly beneficial to the collection performance, as indicated in Fig. 5.7d. However, the effect becomes smaller when most of the data is collected, and the UAVs start to prioritize minimizing overall flight time and safe landing over the collection of the last bits of data.

It is further shown in Fig. 5.7 how the agents react to scenario parameters which were not encountered during training. The corresponding values are highlighted with dashed lines. It can be seen that the performance of the agents follows the same trend as in the rest of the data, when increasing the number of devices (Fig. 5.7b) or initial data per device (Fig. 5.7c) out of the trained region. When increasing the maximum flying time (Fig. 5.7d) for the Manhattan32 agents, or decreasing it for Urban50 agents, the collection ratio with successful landing performance, increases or decreases accordingly. Incrementing the number of agents to four (Fig. 5.7a) reduces the performance slightly. The reason is the decrease in landing performance. However, this is to be expected since the probability of all agents landing decreases with the number of agents. Since the collection ratio is nearly saturated for the scenarios with three agents, the drop in overall

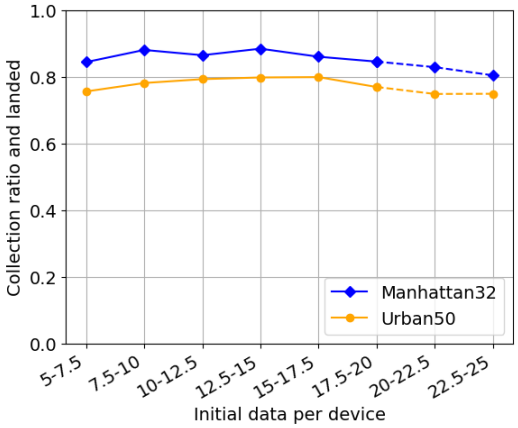


(a) Number of agents $I \in \{1, 2, 3, 4\}$

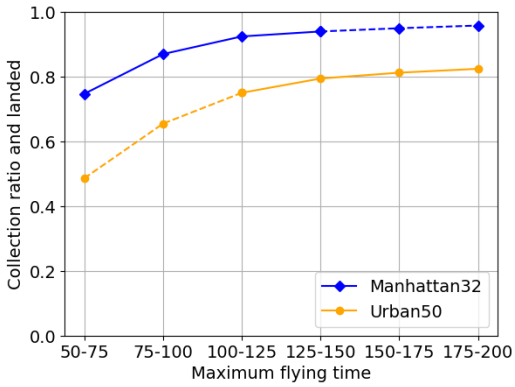


(b) Number of devices $K \in [3, 14]$ sorted into bins of two.

.



(c) Data to be collected from devices $D_{k,init} \in [5, 25]$ sorted into eight bins.



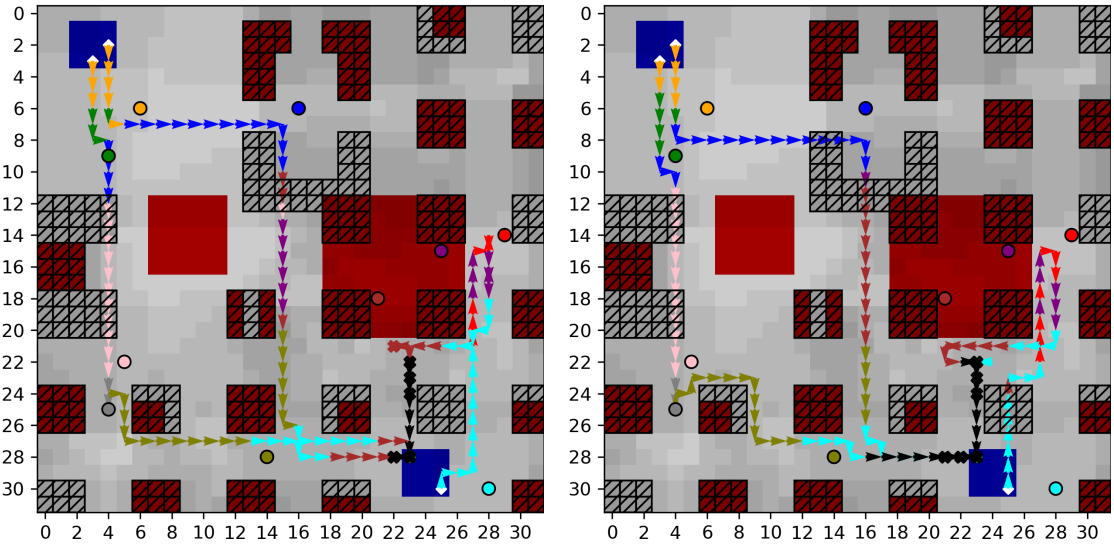
(d) Maximum flying time sorted into six bins in $b_0 \in [50, 200]$

Figure 5.7 – Influence of specific scenario parameters on the data collection ratio with successful landing of all agents. Each data point is an average of 500 Monte Carlo iterations over the respective parameter spaces for the 'Manhattan32' and 'Urban50' map. The parameters within the training range are rendered in solid lines and the out-of-distribution parameter evaluation in dashed lines.

landing performance decreases the collection ratio and landed performance. In general, it is evident that the proposed approach cannot only generalize over the whole range of scenario parameters encountered during training but can also extrapolate successfully to out-of-distribution parameters.

5.6.6 Discussion of the Algorithm's Dependency on the Channel Model

Link Performance Model



(a) $\alpha_{\text{LoS}} = 2.27$, $\alpha_{\text{NLoS}} = 3.64$ (b) $\alpha_{\text{LoS}} = 2.1$, $\alpha_{\text{NLoS}} = 3.0$

Figure 5.8 – Trajectory plots illustrating the influence that a change in propagation conditions has on the already trained agent. Fig. 5.8a shows the original behavior with path loss exponents at the level the agent was trained with. Fig. 5.8b shows the same agent in an otherwise unchanged scenario with slightly lower path loss exponents.

In this section, we would like to expand our discussion of the proposed algorithm's relationship with the rather simple channel model. In principle, our algorithm is based on model-free Q-learning and as such does not use any explicit knowledge about the environment the agent interacts with, specifically it does not use the transition probability function of the MDP [83]. Hence, the approach does not rely on any single specific environment model or channel model.

Although our algorithm can be simply retrained in a scenario with any type of channel model, we are aiming in this work for something else: to provide an algorithm that is able to generalize over scenario parameters, therefore does not require retraining when the scenario changes. However, we did not consider the channel model as part of the variable mission scenario parameters here. Consequently, it can be expected that the agent would need to be retrained when the channel model changes significantly to achieve the best possible data collection performance.

This is put into perspective by the fact that the UAV agents mostly exploit two features of the channel model to make movement decisions. The first one is the simple fact that the smaller the UAV to IoT device distance, the higher the achievable data rate and the more data can be collected. Secondly, the strong dependence of the data rate on LoS vs. NLoS link conditions in an urban environment entices the UAV to choose trajectories that maximize LoS connection opportunities. As both features are very fundamental and would be present in any urban channel model, it is plausible to assume that the agent would perform reasonably well even without retraining in a new channel model. Also recall that the agent has no prior knowledge of the radio map, therefore has to build an internal representation of LoS/NLoS positions from the known building and IoT device positions during the learning process.

To illustrate how much a simple change in the channel model influences the behavior of the agent without retraining, we have replotted the scenario from Fig. 5.5c. For the otherwise unchanged scenario in Fig. 5.8b, we have set the path loss exponents below the levels that were used during the agent's training process. For the most part, the behavior does not differ significantly, but some smaller deviations in the trajectories compared to Fig. 5.8a are recognizable. Making the propagation environment slightly more favorable changes the UAV agents' control decision in so far as the collected and remaining data evolve differently over the course of the mission. The reduction in path loss means that, e.g. at the mission time half point, there is already more data collected leading the agents to adapt their immediate behavior accordingly. As the agents have otherwise no means to detect the change in the channel model, their overall strategy will not change. This of course in the case without retraining or explicitly trying to generalize the learning over different channel coefficients.

Multiple Access Protocol

As described in section 5.2.2, the TDMA protocol ensures that no inter-UAV interference exists in our model and synchronization between the different UAVs is not necessary. We also assume that the IoT devices are capable of simultaneously communicating with all UAVs on all orthogonal frequency bands. This is a simplification owing to the focus of this work on demonstrating multi-UAV, map-based and generalizing trajectory planning. An extension of this work could be envisioned where scheduling decisions are included in the action space of the UAV agents and optimized.

Designing a MAC protocol for aerial networks in general is a challenging research problem in itself due to high mobility and frequent link quality changes. Various UAV MAC protocols have been proposed, considering different access mechanisms (TDMA, CDMA, NOMA, etc.), as well as omni-directional or beamforming antennas [44]. We would like to refer to a specific proposed UAV MAC protocol [121] that utilizes itself a distributed Q-learning scheme to switch between TDMA and CSMA/CA at the UAV depending on which is more appropriate for the current state of the UAV. In [121], the mentioned problem of synchronization is solved through a synchronous switching procedure based on practical byzantine fault tolerance (PBFT) consensus decision. Integration of such schemes is in principle possible with our proposed trajectory optimization approach and

could be explored in future work, but also increases the system complexity significantly.

It is possible that a change of the multiple access protocol, e.g. introducing inter-UAV interference, could trigger a larger change in what constitutes the optimal behavior for multiple UAVs, which consequently might require retraining. However, referring to Fig. 5.6c a natural behavior emerges that would actually mitigate the influence of inter-UAV interference. It is clear, that the three UAV agents tend to divide the data collection task geographically among themselves. This increases the average inter-UAV distance over the mission and would mitigate the importance of interference to some extent in any case.

5.6.7 Comparison of UAV-aided and Stationary Base Station System

Having analyzed the DRL multi-UAV data harvesting system in detail, we now compare the data collection performance of the UAV system to a traditional stationary base station on the 'Manhattan32' map. To this end, we place a standard tri-sector base station right at the center of the map, as depicted in Fig. 5.9. For simplification, we assume that the antennas are mounted at a height of 10m, identical to the UAV altitude. The BS has the advantage of being able to use directional antennas with higher gain, compared to the smaller omnidirectional AP antenna on board the UAVs. Comparing the specifications of the antenna that is used on experimental UAV APs at EURECOM [122] and a standard LTE base station antenna [123, 124], we roughly estimate that the stationary base station achieves a 10dB higher gain on average at frequencies around 2GHz. For the sake of a fair comparison, all other parameters like MAC protocol and available frequency resources are assumed to be identical.

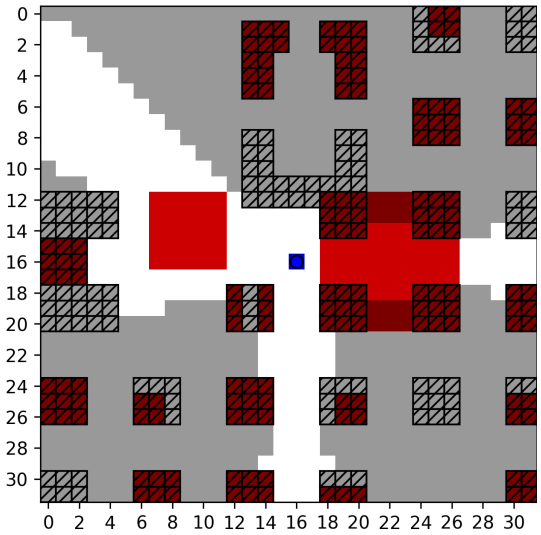


Figure 5.9 – 'Manhattan32' map with base station placed at the center and marked in blue, showing the positions with LoS connectivity in white and NLoS connectivity in gray.

	Base Station	Single UAV	Two UAVs	Three UAVs
Collection Ratio	34.5%	74.8%	92.7%	97.4%

Table 5.5 – Data collection ratio of stationary base station at the map center vs. UAV data harvesters on the 'Manhattan32' map, each averaged over 1000 random scenario Monte Carlo iterations.

Tab. 5.5 lists the data collection ratios, each averaged over 1000 random scenario instances, for the base station at the center of the map and for $I \in \{1, 2, 3\}$ UAV data harvesters. Despite the advantage in antenna gain, a single UAV is on average able to collect twice as much data as the BS. Increasing the number of UAVs leads to nearly complete data collection on average. Note with the exception of the number of UAVs, all other scenario parameters are still sampled randomly from the same value ranges as in section 5.6.3. This also explains why the BS is able on average to collect 34.5% of the data: in some scenarios where the IoT devices are placed by chance close to the BS, it is able to collect the data effectively, while in other scenarios where devices are placed behind buildings, the BS will almost collect no data at all. It is also important to consider that the relatively smaller size of the 'Manhattan32' map, as well as the fact that a BS placed at the center on this map can establish LoS connections with devices for significant parts of the map (e.g. the open area at the top left corner), work in the favor of the BS. The advantage of the UAV-aided system would increase on a larger map with more evenly distributed obstacles like the 'Urban50' map.

5.6.8 Inter-UAV Interference

We are concluding the results section with a short investigation into the effects that interference has on the performance of the multi-UAV data harvesting system. Specifically, we are now assuming that UAVs do not operate on orthogonal resource blocks, but share the same spectral resources, and that consequently, inter-UAV interference exists in our model (analogous to intercell interference). We analyze the effects on the smaller 'Manhattan32' map and without considering any interference-mitigating scheduling strategy. Therefore, the effects on performance are expected to be severe.

Consequently and in contrast to equation (5.8), we now have the signal-to-interference-plus-noise ratio (SINR) at time n with transmit power $P_{i,k}$, noise power N, channel gain $h_{i,k}(n)$ between the ith UAV and kth device given as

$$\mathrm{SINR}_{i,k}(n) = \frac{P_{i,k} h_{i,k}(n)}{N + I_{i,k}(n)}. \tag{5.37}$$

The channel gain $h_{i,k}(n)$ is otherwise identical to the one in the previous SNR formulation from equation (5.8). The total interference power at UAV i at time n is given by

$$\mathrm{I}_{i,k}(n) = \sum_{\forall l \neq k} P_{i,l} h_{i,l}(n). \tag{5.38}$$

Metric	No interference	With interference (no retraining)	With interference and retraining
Successful Landing	99.4%	92.4%	97.6%
Collection Ratio	88.0%	58.5%	67.7%
Collection Ratio and Landed	87.5%	55.3%	66.0%

Table 5.6 – Interference performance metrics on the 'Manhattan32' map averaged over 1000 random scenario Monte Carlo iterations.

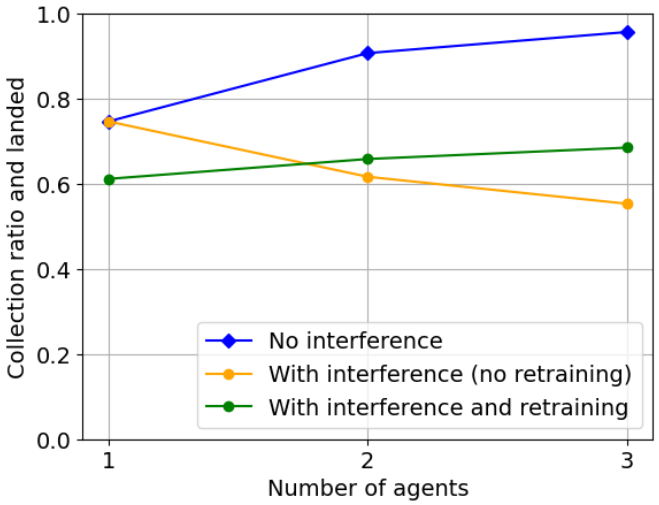


Figure 5.10 – Influence of interference on the collection ratio with successful landing performance metric depending on the number of deployed UAVs. Each data point is the average of 500 random Monte Carlo scenarios.

In Tab. 5.6, we present the overall performance figures for the original case with no interference, the case where we evaluate the agent in an environment with interference that was previously trained exclusively in environments without interference (without any retraining), and the case where we train the DRL system from the beginning on environments with interference. In Fig. 5.10, the performance in the same three cases is shown depending on the number of deployed UAVs. As expected, simply evaluating the previously trained agents on scenarios that include interference, leads to large performance losses as it is impossible for the DRL system to adapt to such a fundamental change in the underlying reward calculation without retraining. Adding additional UAVs in this case actually leads to steeply reduced performance. However, it is clear that when the system is trained from the beginning in scenarios that include inter-UAV interference, overall performance increases and adding additional UAVs has again a positive effect on the data collection performance. As scheduling decisions are not part of the UAV agent action space here and otherwise no interference-mitigating strategies are used, this

signifies that the agents learn to optimize their movement decision with respect to the interference to some degree, e.g. by avoiding to move to positions with LoS conditions for other active UAV-device links that would cause strong interference, basically the UAV agents are 'hiding' from each other behind buildings. For future work, it would be of course advantageous to consider interference-mitigating scheduling strategies [44, 121] or to make scheduling decisions part of the action space and jointly optimizing them with path planning decisions [125, 126].

5.7 Conclusion

We have introduced a multi-agent reinforcement learning approach that allows us to control a team of cooperative UAVs on a data harvesting mission in a large variety of scenarios without the need for recomputation or retraining when the scenario changes. By leveraging a DDQN with combined experience replay and convolutionally processing dual global-local map information centered on the agents' respective positions, the UAVs are able to find efficient trajectories that balance data collection with safety and navigation constraints without any prior knowledge of the challenging wireless channel characteristics in the urban environments. We have also presented a detailed description of the underlying path planning problem and its translation to a decentralized partially observable Markov decision process. Planning a trajectory for UAVs in the scenario of a UAV BS (chapter 3) and the (multi)-UAV data harvesting from IoT devices (this chapter and chapter 4), are very similar problems that essentially can be solved by means of the same DRL methods. In the following chapter, we will demonstrate that also other UAV path planning problems can be tackled directly by the methods that were introduced. By finding a common map-based description of the IoT data collection and the coverage path planning (CPP) problem, we show that the special flexibility of the DRL paradigm allows us to solve both problems from distinctly different research communities, robotics and UAV-aided communications, with identical methodology.

Chapter 6

Coverage Path Planning

6.1 Introduction

The preceding chapters have demonstrated how DRL methods can be applied to various scenarios of UAV trajectory planning in UAV-aided communication networks. This chapter is a little excursion into the adjacent research area of aerial robotics. Despite working on highly similar problems, as will be demonstrated in this chapter, the two research communities of UAV communications and aerial robotics are often disjoint and research runs mostly along parallel, non-intersecting lines. This chapter aims to give some insight into possible synergies. For ease of exposition, we analyze both problems in the single UAV case.

Coverage path planning (CPP) is the task of designing a trajectory that enables a mobile agent to travel over every point of an area of interest. Whereas the CPP problem for ground-based robotics has already found its way into our everyday life in the form of vacuum cleaning robots [127], autonomous coverage with UAVs, while not yet having attained the same level of prominence, is being considered for a wide range of applications, such as photogrammetry, smart farming and especially disaster management [128]. UAVs can for example be deployed rapidly to gather initial or continuous survey data of areas hit by natural disasters, or mitigate their consequences. In the aftermath of the 2019-20 Australian bushfire season, wildlife officers inventively used quadcopter drones with infrared sensors to conduct a search-and-rescue operation for koalas affected by the blaze [129].

As its name suggests, covering all points inside an area of interest with CPP is related to conventional path planning where the goal is to find a path between start and goal positions. In general, CPP aims to cover as much of the target area as possible within given energy or path-length constraints while avoiding obstacles or no-fly zones. The same power constraints as in UAV-aided communications lead to similar constraints on flying time. Finding a CPP control policy that generalizes over varying power constraints and setting a specific movement budget can be seen as a way to model the variations in actual power consumption during the mission, e.g. caused by environmental factors that are hard to predict. Similar to conventional path planning, CPP can usually be reduced to some form of the traveling salesman problem, which is NP-hard [127]. Lawn-mowing

and milling [130] are other examples of closely related problems.

6.1.1 Related Work

To guarantee complete coverage, most classical CPP approaches split the target area and surrounding free space into cells, by means of exact or approximate cellular decomposition. Choset and Pignon [131] proposed the classical boustrophedon ("the way of the ox", back and forth motion) cellular decomposition, an exact decomposition method that guarantees full coverage but offers no bounds on path-length. This algorithm was extended by Mannadiar and Rekleitis [132] through encoding the cells of the boustrophedon decomposition as a Reeb graph and then constructing the Euler tour that covers every edge in the graph exactly once. Cases where the mobile agent does not have enough power to cover the whole area are not considered. The authors in [133] adapted this method for use in a non-holonomic, fixed-wing UAV and conducted extensive experimental validation. Two other approaches combining CPP and the traveling salesman problem to find near-optimal solutions for coverage of target regions enclosed by non-target areas are proposed by the authors in [134]: grid-based and dynamic programming-based, respectively. Both approaches suffer from exponential increase in time complexity with the number of target regions and do not consider obstacles or UAV power limitations.

Non-standard approaches have made use of neural networks (NNs) before. The authors in [135] design a network of neurons with only lateral connections that each represent one grid cell in a cleaning robot's non-stationary 2D environment. The path planning is directly based on the neural network's activity landscape, which is computationally simple and can support changing environments, but does not take path-length or power constraints into account.

Reinforcement learning with deep neural networks has only recently started to be considered for aerial robotics path planning. Maciel-Pearson *et al.* [107] proposed a method using an extended double deep Q-network (EDDQN) to explore and navigate from a start to a goal position in outdoor environments by combining map and camera information from the drone. Their approach is focused on obstacle avoidance under changing weather conditions. The authors in [136] investigate the CPP-related drone patrolling problem where a UAV patrols an area optimizing the relevance of its observations through the use of a single-channel relevance map fed into a convolutional layer of a DDQN agent. However, there is no consideration for power constraints and the relevance map is preprocessed showing only local information.

Other UAV path planning works exist that also made use of convolutional map processing for DRL agents. In [137], fixed-wing UAVs are tasked with monitoring a wildfire propagating stochastically over time. Control decisions are based on either direct observations or belief maps fed into the DRL agents. The focus here is the inherent uncertainty of the problem, not balancing a mission goal and navigation constraints in large complex environments. Wildfire surveillance is also the mission of the quadcopter UAVs in [138], which is set in a similar scenario without navigation constraints and makes use of uncertainty maps to guide path planning. Their approach is based on an extended Kalman filter and not the reinforcement learning (RL) paradigm. To monitor another

natural disaster situation, Baldazo *et al.* [139] present a multi-agent DRL method for flood surveillance using the UAVs' local observations of the inundation map to make control decisions. All mentioned approaches focus on solving a single class of UAV missions in simple physical environments and do not consider combining local and global map information.

A recent survey of UAV coverage path planning is given by Cabreira *et al.* [128]. Galceran and Carreras [127] provide a survey of general (ground robotics) approaches to CPP. A survey of various applications for UAV systems from a cyber-physical perspective is offered in [74].

6.1.2 Contributions

In this chapter, we aim to establish the previously described DDQN approach with global-local map processing as a general method for UAV trajectory planning by demonstrating its applicability to two distinctly different mission scenarios: coverage path planning (CPP) and path planning for wireless data harvesting (DH). Control policy generalization over scenario parameters is extended to fully randomly generated target zones in coverage path planning. We also analyze and discuss the effects of key map processing parameters on the trajectory learning performance. This work also establishes DRL as a promising approach to the UAV CPP problem.

6.2 Problem Formulation

6.2.1 Agent System

The sensors of the CPP UAV forming the input of the reinforcement learning agent are depicted in Figure 6.1: camera and GPS receiver. The camera gives a periodic frame of the current coverage view and the GPS yields the drone's position. Power constraints determined by external factors are modeled as a movement budget for the drone that is fixed at mission start. Two additional software components are running on the UAV. The first is the mission algorithm which is responsible for the analysis of the camera data. We assume that any mission algorithm can give feedback on the area that was already covered. The second component is a safety controller (see (5.15)) that evaluates the proposed action of the agent and accepts or rejects it based on the safety constraints (entering into no-fly zones or landing in unsuitable areas). Note that the safety controller does not assist the agent in finding the landing area. The last component is a map which is provided by the operator on the ground. While this map could be dynamic throughout the mission, we focus on static maps for the duration of one mission here.

6.2.2 Environment and UAV Model

We consider the same model as in previous chapters: a grid world of size $M \times M \in \mathbb{N}^2$ with cell size c. The environment contains designated start/landing positions, regulatory no-fly zones (NFZs), and obstacles. The map can be described through a tensor $\mathbf{M} \in \mathbb{B}^{M \times M \times 3}$, where $\mathbb{B} = \{0, 1\}$ and with the start/landing zones in map-layer 1, the union of NFZs

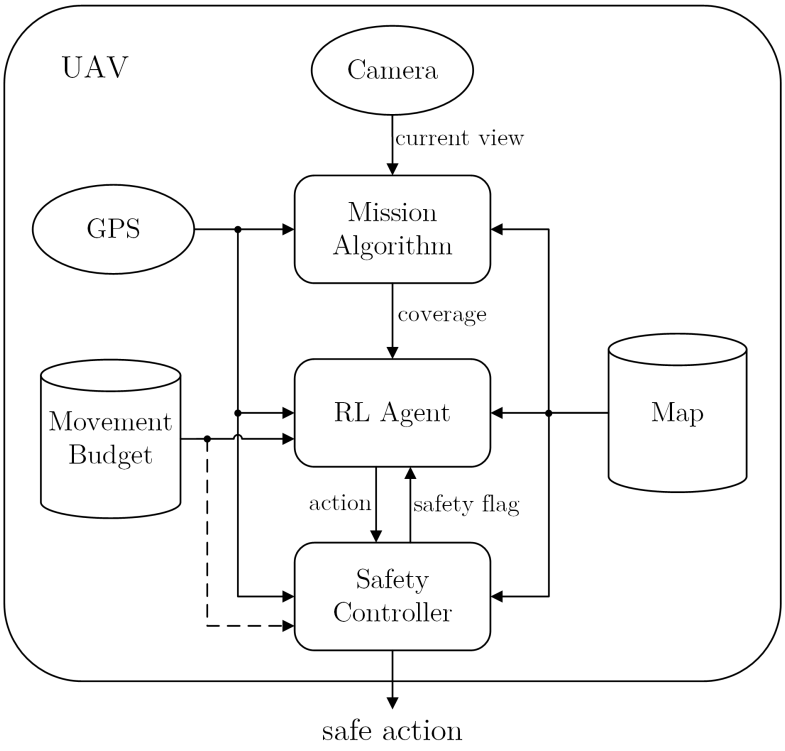


Figure 6.1 – System diagram.

and obstacles in map-layer 2, and the obstacles alone in map-layer 3. The UAV position is defined through $\mathbf{p}(t) \in \mathbb{N}^2$.

6.2.3 Target and Mission Definitions

In the following, we show that a universal description for coverage path planning (CPP) and data harvesting (DH) can be found through separation into two parts, the environment and the target.

Coverage Path Planning

In coverage path planning, the mission is to cover a designated target area by flying above or near it, such that it is in the field of view of a camera-like sensor mounted underneath the UAV. The target area can be described through $\mathbf{T}(t) \in \mathbb{B}^{M \times M}$, in which each element describes whether a cell has to be covered or not. The current field of view of the camera can be described with $\mathbf{V}(t) \in \mathbb{B}^{M \times M}$ indicating for each cell whether it is in the current field of view or not. In this work, the field of view is a square of 5×5 surrounding the current UAV position. Additionally, buildings can block line-of-sight, which is also incorporated in calculating $\mathbf{V}(t)$. This prohibits the UAV from 'seeing' around the corner.

Consequently, the target area evolves according to

$$\mathbf{T}(t+1) = \mathbf{T}(t) \wedge \neg \mathbf{V}(t), \tag{6.1}$$

in which $\wedge$ and $\neg$ are the cell-wise logical *and* and *negation* operators, respectively. In our mission definition, obstacle cells in the environment cannot be a coverage target, while start and landing zones and no-fly-zones can be. The goal is to cover as much of the target area as possible within the maximum flying time constraint.

Data Harvesting

Conversely, the mission in path planning as stated for wireless data harvesting in chapter 4 is to collect data from $K \in \mathbb{N}$ stationary IoT devices spread throughout the environment at ground-level, with the position of device $k \in [1, K]$ given through $\mathbf{u}_k \in \mathbb{N}^2$. Each device has an amount of data $D_k(t) \in \mathbb{R}$ to be collected by the UAV. Link performance and MAC protocol are identical with section 5.2.2. Hence, the data at each device evolves according to

$$D_k(t+1) = D_k(t) - C_k(t) \tag{6.2}$$

with data throughput $C_k(t)$ (5.11). Devices can be located in every cell except for the starting and landing zones or inside obstacles.

Unifying Map-Layer Description

Both problems can be described through a single target map-layer $\mathbf{D}(t) \in \mathbb{R}^{M \times M}$. In CPP, the target map-layer is given through $\mathbf{T}(t)$ evolving according to (6.1). In DH, the target map-layer shows the amount of available data in each cell that one of the devices is occupying, i.e. the cell at position $\mathbf{u}_k$ has value $D_k(t)$ and is evolving according to (6.2). If a cell does not contain a device or the device data has been collected fully, the cell's value is 0. Since the two problems can be described with similar state representations, both can be solved through deep reinforcement learning with a neural network having the same structure.

6.3 Methodology

To address both problems we formulate them as a partially observable Markov decision process (POMDP) [140] which is defined through the tuple $(\mathcal{S}, \mathcal{A}, P, R, \Omega, \mathcal{O}, \gamma)$. Decentralization is not necessary for the single UAV case, but otherwise the definitions from section 5.3 are also valid here.

We unify the UAV path planning problems by describing their state space with

$$\mathcal{S} = \underbrace{\mathbb{B}^{M \times M \times 3}}_{\substack{\text{Environment} \\ \text{Map}}} \times \underbrace{\mathbb{R}^{M \times M}}_{\substack{\text{Target} \\ \text{Map}}} \times \underbrace{\mathbb{N}^2}_{\text{Position}} \times \underbrace{\mathbb{N}}_{\substack{\text{Flying} \\ \text{Time}}}, \tag{6.3}$$

in which the elements $s(t) \in \mathcal{S}$ are

$$s(t) = (\mathbf{M}, \mathbf{D}(t), \mathbf{p}(t), b(t)). \tag{6.4}$$

The four components of the tuple are

- $\mathbf{M}$ the environment map containing start and landing zones, no-fly zones, and obstacles;
- $\mathbf{D}(t)$ the target map indicating remaining data at device locations or remaining cells to be uncovered at time t;
- $\mathbf{p}(t)$ the UAV's position at time t;
- $b(t)$ the UAV's remaining movement budget at time t;

The UAV's action space is identical to (4.3). The generalized reward function $R(s(t), a(t), s(t+1))$ consists of the following elements:

- r_c *(positive)* the data collection or cell covering reward given by the collected data or the amount of newly covered target cells, comparing $s(t+1)$ and $s(t)$;
- r_{sc} *(negative)* safety controller (SC) penalty in case the drone has to be prevented from colliding with a building or entering an NFZ;
- r_{mov} *(negative)* constant movement penalty that is applied for every action the UAV takes without completing the mission;
- r_{crash} *(negative)* penalty in case the drone's remaining flying time reaches zero without having landed safely in a landing zone.

The neural network model for the DDQN approach is identical to Fig. 5.3, with the exception of different sizes for the convolutional and flatten layers depending on the factors chosen for the global-local map processing, which was described in section 5.4 previously. The loss function for the DDQN training is given in (4.4).

As explained in section 5.4, the relevant parameter for scalability to larger maps is the size of the flatten layer. It can be calculated through

$$N = n_k \left(\left(l - n_c \lfloor \frac{s_k}{2} \rfloor \right)^2 + \left(\lfloor \frac{M_c}{g} \rfloor - n_c \lfloor \frac{s_k}{2} \rfloor \right)^2 \right) + 1 \tag{6.5}$$

with n_k being the number of kernels, n_c the number of convolutional layers, and s_k being the kernel size. The standard hyperparameters for 32×32 and 50×50 maps are listed in Tab. 5.2. Setting the global map scaling parameter to $g = 1$ and the local map size to $l = 0$ deactivates global-local map processing, i.e., no downsampling and no extra local map.

6.4 Simulations

6.4.1 Simulation Setup

We use again the two different grid worlds from the previous chapter 5: the 'Manhattan32' scenario the 'Urban50' scenario. The cell size for the scenarios is 10m $\times$ 10m with Table 6.1 providing the legend for plots.

For the CPP problem, the UAV is flying at a constant altitude of 25m with a camera mounted underneath that has a field of view angle of 90°. Consequently, the UAV can cover an area of 5×5 cells simultaneously, as long as obstacles do not block line of sight. The target areas are generated by randomly sampling geometric shapes of different sizes and types and overlaying them, creating partially connected target zones. For evaluation, a traditional metric for the CPP problem is the path length. However, this metric only offers meaningful comparison when full coverage is possible. In this work, we investigate flight time constrained CPP, in which full coverage is often impossible. Therefore, the evaluation metrics used are the coverage ratio (CR), i.e. the ratio of covered target cells to the initial target cells at the end of the episode, and coverage ratio and landed (CRAL), which is zero if the UAV did not land successfully and equal to CR if it did. The benefits of the CRAL metric are that it combines the two goals, achieving high coverage and returning to the landing zone within the flight time constraint. By normalizing performance to a value in $[0, 1]$, it enables performance comparisons over the changing scenarios with randomly generated target zones. Evaluation in the DH scenario was already explained in section 4.6.1 and uses the data collection ratio instead of the coverage ratio. The abbreviations CR and CRAL either stand for coverage or collection ratio depending on the context mission.

Symbol	Description
DQN Input	
	Start and landing zone
	Regulatory no-fly zone (NFZ)
	Buildings blocking wireless links and FoV
	Remaining target zone (yellow also NFZ)
Visualization	
	Not covered and covered
	Starting and landing positions during an episode

Table 6.1 – Legend for CPP scenario plots.

6.4.2 General Evaluation

The CPP problem agents were trained on target zones containing 3-8 shapes covering 20-50% of the available area. The movement range was set to 50-150 steps for the 'Manhattan32' scenario and 150-250 for the 'Urban50' scenario. For the DH scenarios, 3-10 devices are placed randomly in free cells and contain 5.0-20.0 data units. The movement range was set to 50-150 steps for the 'Manhattan32' scenario and 100-200 for the 'Urban50' scenario. Four scenarios are evaluated in detail.

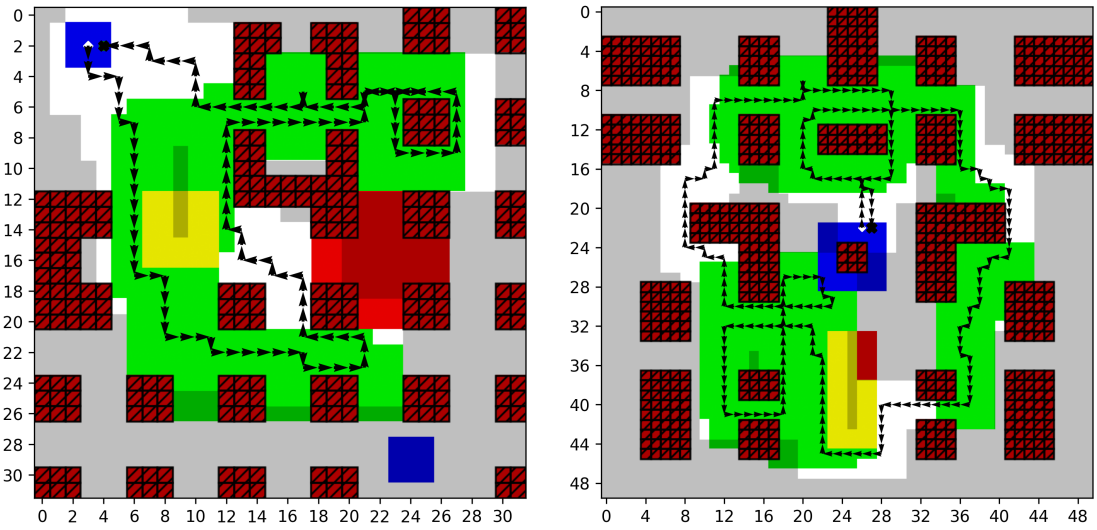
In the CPP scenarios, the agents in Fig. 6.2a and 6.2b show that they can find trajectories to cover most of the target area. Even the area in the NFZs is mostly covered. It can be seen that small areas that would require a detour are ignored, leading to incomplete coverage. However, most of the target area is covered efficiently. Plots for

Metric	Manhattan32 CPP	Manhattan32 DH	Urban50 CPP	Urban50 DH
Landed	98.5%	98.2%	98.1%	99.5%
CR	71.0%	83.6%	81.5%	74.5%
CRAL	70.3%	82.5%	80.1%	74.2%

Table 6.2 – Performance metrics averaged over 1000 random scenario Monte Carlo iterations.

the DH scenario were already shown in Fig. 5.5 and 5.6, although separate agents were trained here to enable direct comparison.

All four agents (one for each map and the CPP or DH problem) were trained for 2 million steps. When analyzing their performance in all four missions using 1000 Monte Carlo generated scenarios (see Table 6.2), it can be seen that all agents' landing performances are good, with the 'Urban50' DH agent being slightly better.



(a) Movement 124/140, coverage ratio 0.94

(b) Movement 234/250, coverage ratio 0.94

Figure 6.2 – Example trajectories from the Monte Carlo simulations for CPP on 32×32 Manhattan map and 50×50 Urban map.

6.4.3 Global-Local Parameter Evaluation

To establish the performance sensitivity to the new hyper-parameters, global map scaling g, and local map size l, we trained multiple agents with different parameters on the CPP and DH problems. We chose four values for l and four for g and trained three agents for each possible combination. Additionally, we trained three agents without the usage of global and local map processing, which is equivalent to setting $g = 1$ and $l = 0$.

Global map scaling g	Local map scaling l			
	9	17	25	33
2	8,481	9,761	13,089	18,465
3	2,721	4,001	7,329	12,705
5	273	1,553	4,881	10,257
7	33	1,313	4,641	10,017

Table 6.3 – Flatten layer size for 'Manhattan32' with different global map scaling and local map sizes; Without global-local map processing the size is 48,401 neurons.

Global map scaling g	Local map scaling l							
	9		17		25		33	
	CPP	DH	CPP	DH	CPP	DH	CPP	DH
2	2.7	2.2	2.3	2.0	1.8	1.6	1.3	1.1
3	3.5	3.0	3.0	2.5	2.2	1.9	1.6	1.4
5	4.2	3.6	3.4	3.0	2.5	2.2	1.9	1.6
7	4.7	3.8	3.6	3.0	2.5	2.2	2.5	2.1

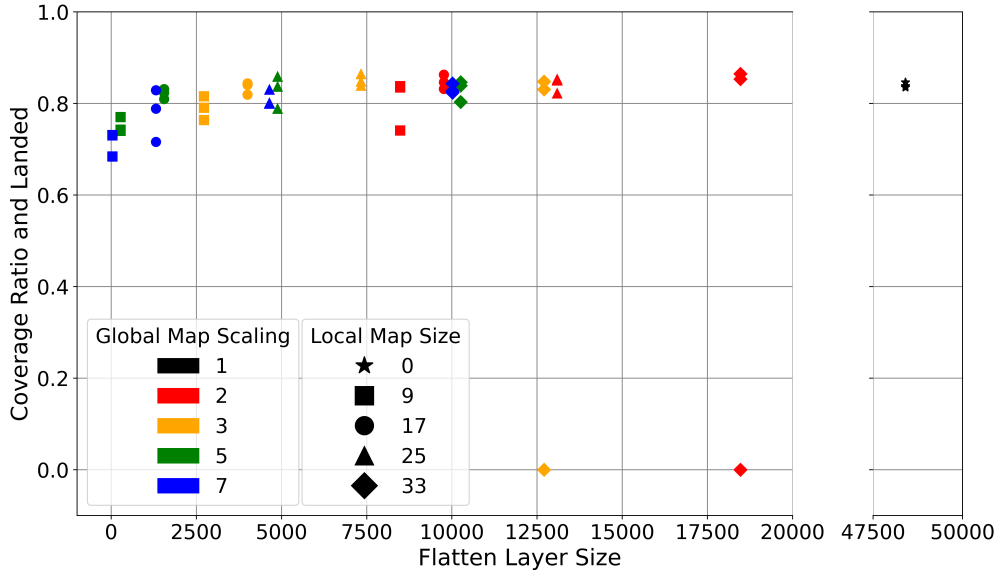
Table 6.4 – Training time speedup for the CPP and DH problem relative to without global-local map processing.

The resulting 51 agents for the CPP and DH problems were trained for 500k steps each and evaluated on 200 Monte Carlo generated scenarios. The difference to the previous evaluation is that the movement budget range was set to $150 - 300$.

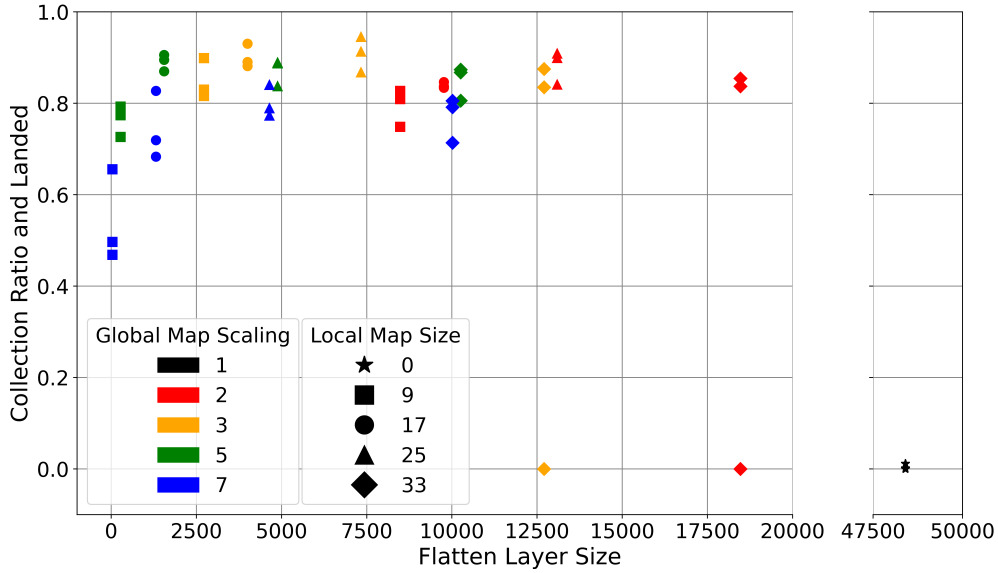
Table 6.3 shows the selected parameters and the resulting flatten layer size according to (6.5). A significant speedup of the training process compared to agents without global and local map processing can be observed in Table 6.4.

The resulting CRAL values from the Monte Carlo simulations for each agent with respect to the agent's flatten layer size are shown in Fig. 6.3a and 6.3b for the CPP and DH problem, respectively. It can be seen that the DH problem is more sensitive to the parameters than the CPP problem. Generally, a larger flatten layer yields better performance up to a point. For both problems, it can be seen that a large flatten layer can cause the learning to get unstable, resulting in a CRAL of zero for some runs. This is caused by the agent's failure to learn how to land. The DH agents, which are not using the global-local map approach, never learn how to land reliably and thus have a CRAL score near zero.

In both cases, the agents with $l = 17$ and $g = 3$ or $g = 5$ show the best performance with respect to their flatten layer size, justifying the selection in Table 5.2. Besides these two parameter combinations, it is noteworthy that the agents with $l = 9$ and $g = 7$ also perform well in both scenarios, despite their small flatten layer size of only 33 neurons.



(a) Grid search for CPP



(b) Grid search for DH

Figure 6.3 – Parameter grid search for CPP and DH with parameters from Table 6.3; the black stars correspond to agents without global-local map processing.

6.5 Conclusion

We have presented a method for generalizing autonomous UAV control over two distinctly different mission types, coverage path planning and data harvesting. Through the flexibility afforded by combining specific mission goals and navigation constraints in the reward function, we trained DDQNs with identical structures in both scenarios to make efficient movement decisions. We have analyzed the effects of the dual global-local map processing parameters on learning performance. After this excursion into the research field of aerial robotics, we will return to a different formulation of the IoT data collection scenario in the next chapter and explore the idea of using a model-accelerated DRL training procedure that reduces the requirement for training data collected in the real-world, however while learning a control policy for single specific scenario at a time.

Chapter 7

Model-aided Sample-efficient UAV Trajectory Planning

7.1 Introduction

In this final chapter, we explore another approach to solve the training data demand challenge of DRL: model-aided DQN learning. At the same time we investigate a data collection scenario in which the UAV agent has more prior knowledge about the environment than in the scenario described in chapter 3, but a little less than in the scenarios of chapters 4 and 5. In contrast to the approaches in chapters 4, 5 and 6, we do not look at generalizing the UAV control policy over many scenario parameter variations here, but focus on solving a single scenario (similar to chapter 3) with the least amount of expensive real-world training data possible. In contrast to the previous scenarios, we also assume that the UAV flies above the city buildings at all times. Hence, obstacle avoidance is not a concern in this trajectory planning problem. In contrast to the chapters 4 and 5, but in accordance with chapter 3, we also assume that an unlimited amount of data can be picked up from each IoT device. In general, the assumptions in this chapter are closer to the ones in chapter 3, in order to enable a comparison of the model-aided approach with the baseline of deep Q-learning with scalar state space input and no prior information.

The model-aided deep Q-learning approach considerably reduces the need for expensive training data samples, while still achieving the overarching goal of DRL, i.e to guide a battery-limited UAV towards an efficient data harvesting trajectory, without prior knowledge of wireless channel characteristics and limited knowledge of wireless node locations. On the one hand, approaches as the one from chapter 3 assume absolutely no prior knowledge of the environment, but require large amounts of training data even in a simple environment as the DRL agent has to deduce the scenario conditions purely by trial and error. On the other hand, near perfect state information in works such as [112], where cooperative UAVs are tasked with collecting data from IoT devices in a relatively simple unobstructed environment, enables faster convergence and requires less training data. Here, prior knowledge available to the UAV agent is in between the two

extremes: while some reference IoT node positions are known (referred to as anchors), other node positions and the challenging wireless channel characteristics in a dense urban environment that causes alternation between LoS and NLoS links, must be estimated and a model of the environment constructed. Offline interaction with the model allows us to train a DQN to approximate the optimal UAV control policy. In comparison with standard DRL approaches, the proposed model-aided approach requires at least one order of magnitude less training data samples to reach identical data collection performance.

In the context of sample-efficient RL, model-accelerated solutions have been proposed previously for a variety of applications. A method called *imagination rollouts* to increase sample-efficiency for a continuous Q-learning variant has been suggested for simulated robotic tasks in [141]. Their approach is based on using iteratively refitted time-varying linear models, in contrast to a DQN model that we propose here. Learning a NN model in the context of stochastic value gradient learning methods has been proposed in [142].

Other works in the area of RL trajectory optimization for UAV communications have suggested other ways of reducing training data demand. Li *et al.* [143] proposed a DRL method for sum-rate maximization from moving users based on transfer learning to reduce training time. In [30], the authors propose meta-learning on random user uplink access demands for distributed UAV BS control, reducing training time by around 50% compared to standard RL. Another possibility as in chapters 4 and 5 is to directly generalize training over a range of likely scenario parameters, where the agent requires no retraining when scenario parameters change. This is at the cost of longer initial training and it also requires that the change is observable for the agents.

To the best of our knowledge, this is the first work that proposes model-based acceleration of the training process in DRL UAV path planning and also the first one that suggests the use of IoT anchor nodes for localization of IoT devices with unknown locations. By introducing a device localization algorithm that exploits a limited number of reference device positions and a city 3D map, we show that our proposed method offers fast convergence even under uncertainty about device positions and without prior knowledge of the challenging radio channel conditions in a dense urban environment. Finally, we compare our model-aided approach to the baselines of standard DRL without any prior information (along the principles of chapter 3) as well as map-based full knowledge DRL (compare chapters 4 and 5) and show that the model-aided approach achieves a reduction in training data demand of at least one order of magnitude with identical data collection performance.

7.2 Problem Formulation

We consider a UAV data collection scenario very similar to the one described in section 4.2.1 and to 5.2.1 (in the single UAV case). There are K static ground level nodes (IoT sensors) distributed in an urban area. The k-th ground node, $k \in [1, K]$, is located at $\mathbf{u}_k \in \mathbb{R}^2$. The ground nodes are split into two groups: nodes with known locations $\mathbf{u}_k, k \in \mathcal{U}_{\text{known}}$, and nodes with unknown locations $\mathbf{u}_k, k \in \mathcal{U}_{\text{unknown}}$. Note that the ground level node assumption is not restrictive and the proposed algorithms can in principle be applied to a scenario where the nodes are located in 3D.

The UAV mission lasts for a maximum duration of $T \in \mathbb{N}$ mission time steps for all UAVs, where the time horizon is discretized into equal mission time slots $t \in [0, T]$ of length δ_t seconds. The UAV's position is given by $\mathbf{p}(t) = [x(t), y(t), h]^{\mathrm{T}}$. The action space is identical to (4.3) and the position evolves according to

$$\mathbf{p}(t+1) = \mathbf{p}(t) + \mathbf{a}(t), \qquad \mathbf{a}(t) \in \mathcal{A}. \tag{7.1}$$

The drone's battery content changes according to

$$b(t+1) = \begin{cases} b(t) - 0.5, & \mathbf{a}_n = \text{hover} \\ b(t) - 1, & \text{otherwise,} \end{cases} \tag{7.2}$$

which is a variation on the battery model from chapter 5 that puts a greater emphasis on the advantage of hovering over flying at speed. Given the fact that IoT devices in this chapter have an unlimited amount of data that can be picked up, we expect that hovering is of greater importance in this scenario compared to chapters 4 and 5.

The AG channel model follows the LoS/NLoS segmented, point-to-point channel described in section 2.1.2 with log-distance path loss and shadow fading. The MAC follows the TDMA principle, but in contrast to chapters 4 and 5, we assume here that all ground nodes have an equal communication time access. The UAV's goal is to collect the maximum amount of data over the course of its mission, while reaching the final position $\mathbf{p}(T) = \mathbf{v_F} \in \mathbb{R}^3$ before the battery runs out.

The trajectory planning problem is again formulated as a standard MDP, similar to chapter 4. As we only investigate the single UAV case and do not use global-local map processing because of the relatively small map, the MDP is neither partially observable, nor decentralized.

The reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ of the MDP is comprised of two elements:

$$r(t) = \sum_{k \in \mathcal{K}} C_k(t) + \lambda(t). \tag{7.3}$$

The first term in (7.3) is the instantaneous collected data from all nodes at time step t, and $\lambda(t)$ is a penalty imposed by the safety controller that guarantees the UAV will reach the terminal point $\mathbf{v}_{\mathrm{F}}$. In contrast to chapters 4 and 5, the safety controller here can force the UAV to return to the final position easily as the UAV is flying above all buildings and obstacle avoidance is of no concern. Specifically, the safety controller at each time step computes the shortest trajectory (a minimum set of actions) and the minimum required power for getting to the destination point from the current UAV location, then based on these values it declines or accepts the current action $\mathbf{a}(t)$ chosen by the UAV. If action $\mathbf{a}(t)$ is rejected, a penalty term will be added to the reward function. The shortest trajectory and the minimum required power computed at time step t by the safety controller are denoted by $\mathcal{A}_t^{sc}$ and b_t^{sc}, respectively. Thus, the safety penalty $\lambda(t)$ is given by

$$\lambda(t) = \begin{cases} \lambda, & b(t) \leq b_t^{sc} \\ 0, & \text{otherwise.} \end{cases} \tag{7.4}$$

The action chosen by the UAV at each time step is checked and modified (if necessary) by the safety controller as follows

$$\mathbf{a}(t) = \begin{cases} \mathbf{a}_{t,1}^{sc}, & b(t) \leq b_t^{sc} \wedge \mathbf{a}(t) \notin \mathcal{A}_t^{sc} \\ \mathbf{a}_t, & \text{otherwise,} \end{cases} \tag{7.5}$$

where $\mathbf{a}_{t,1}^{sc}$ is the first element of $\mathcal{A}_t^{sc}$.

7.3 Model-aided Deep Q-learning

To solve the MDP, we again employ deep Q-learning, also as to enable direct comparison of the model-aided approach with the other approaches in this thesis and the literature. Note that our aim is to reduce the real-world training data samples of Q-learning by model-aided acceleration with an *external* model that simulates the environment. Accordingly, the DQN algorithm (see section 2.3.3) is unchanged and follows the standard cycle of interaction between agent and environment to iteratively learn a policy $\pi(s)$. Tackling again the problem of the large training data demand of DRL methods, we propose an algorithm where the agent learns an environment model continuously while collecting real-world measurements. This model is then used by the agent to simulate experiments and supplement the real-world data.

Specifically, the next state s_{t+1} given the current state s_t and action $\mathbf{a}_t$ can be computed from (7.1), (7.2). The reward function (7.3) consists of two parts: the safety penalty, which is known from (7.5), and the instantaneous collected data from the IoT node devices. Therefore we only need to estimate the instantaneous collected data from devices which according to section 2.1.2, is a function of ground node locations and the radio channel model. Hence, the approximation of the reward function boils down to ground node localization and radio channel learning from collected radio measurements.

The problem of simultaneous wireless node localization and channel learning has been studied in previously [144]. In this section, we propose an approach of model-free node localization by leveraging the 3D map of the environment. Akin to [144] and the previous chapters of this thesis, a LoS/NLoS segmented radio channel is assumed. However, in contrast to [144], the goal here is to estimate the radio channel using a model-free method while localizing the ground nodes. We train a neural network to approximate the radio channel model, which is utilized along with a particle swarm optimization (PSO) technique and a 3D map of the city to localize the wireless nodes with unknown positions.

7.3.1 Simultaneous Node Localization and Channel Learning

As the first goal of the UAV is to learn the channel and localize the unknown IoT nodes, we firstly assume that the UAV follows an arbitrary trajectory denoted by $\chi = \{\mathbf{p}(t), t \in [1, T]\}$ for collecting received signal strength (RSS) measurements, where $\mathbf{p}(t)$ represents the UAV's position in the time interval t. We also assume that the UAV collects radio measurements form all K nodes at each location. Let $g_{t,k}$ represent the RSS measurements (in dB scale) obtained from the k-th node by the UAV in interval

t. Assuming a LoS/NLoS segmented pathloss model that is suitable for air-to-ground channels in urban environments with buildings [145], we have

$$g_{t,k}=\begin{cases}\psi_{\boldsymbol{\vartheta}}(d_{t,k},\phi_{t,k},w_{t,k}{=}1)+\eta_{t,k,\mathrm{LoS}} & \text{if LoS}\\ \psi_{\boldsymbol{\vartheta}}(d_{t,k},\phi_{t,k},w_{t,k}{=}0)+\eta_{t,k,\mathrm{NLoS}} & \text{if NLoS,}\end{cases} \tag{7.6}$$

where $d_{t,k}=\|\mathbf{u}_k-\mathbf{p}(\mathbf{t})\|$, and $\phi_{t,k}=\arcsin(\frac{\bar{d}_{t,k}}{d_{t,k}})$ is the elevation angle between the UAV at time step t and node k with $\bar{d}_{t,k}$ representing the ground distance between the ground node and the UAV. $\omega_{t,k}\in\{0,1\}$ is the classification binary variable (yet unknown) indicating whether a measurement falls into the LoS or NLoS category. The function $\psi_{\boldsymbol{\vartheta}}(.)$ is the channel model parameterized by $\boldsymbol{\vartheta}$. Note that, neither function $\psi(.)$ nor parameters $\boldsymbol{\vartheta}$ are known and need to be estimated. $\eta_{t,k,z}$ represents the shadowing effect following a zero-mean Gaussian distribution with variance σ_z^2, which we assume to be known for both segments $z\in\{\mathrm{LoS},\mathrm{NLoS}\}$. The probability distribution of a single measurement in (7.6) is modeled as

$$p(g_{t,k})=(f_{t,k,\mathrm{LoS}})^{w_{t,k}}(f_{t,k,\mathrm{NLoS}})^{(1-w_{t,k})}, \tag{7.7}$$

where $f_{t,k,z}$ has a Gaussian distribution with $\mathcal{N}(\psi_{\boldsymbol{\vartheta}}(d_{t,k},\phi_{t,k},w_{t,k}),\sigma_z^2)$.

Assuming that collected measurements conditioned on the channel and node positions are independent and identically distributed (i.i.d) [145], using (7.7), the negative log-likelihood of measurements leads to

$$\begin{aligned}\mathcal{L}=&\log\left(\frac{\sigma_{\mathrm{LoS}}^2}{\sigma_{\mathrm{NLoS}}^2}\right)\sum_{k=1}^{K}\sum_{t=1}^{T}\omega_{t,k}+\\ &\sum_{k=1}^{K}\sum_{t=1}^{T}\frac{\omega_{t,k}}{\sigma_{\mathrm{LoS}}^2}\left|g_{t,k}-\psi_{\boldsymbol{\vartheta}}(d_{t,k},\phi_{t,k},w_{t,k})\right|^2+\\ &\sum_{k=1}^{K}\sum_{t=1}^{T}\frac{(1-\omega_{t,k})}{\sigma_{\mathrm{NLoS}}^2}\left|g_{t,k}-\psi_{\boldsymbol{\vartheta}}(d_{t,k},\phi_{t,k},w_{t,k})\right|^2.\end{aligned} \tag{7.8}$$

The estimate of $\psi(.),\boldsymbol{\vartheta}$, and $\mathbf{u_k}$ can then be obtained by solving

$$\min_{\substack{\omega_{t,k},\mathbf{u}_k,\forall t,\forall k\\ \psi(.),\boldsymbol{\vartheta}}}\quad \mathcal{L} \tag{7.9a}$$

$$\text{s.t.}\quad \omega_{t,k}\in\{0,1\},\forall t,\forall k. \tag{7.9b}$$

The binary variables $\omega_{t,k}$ in objective function (7.8), and the fact that $\psi(.)$ is not explicitly known and is a nonlinear function of node locations, make problem (7.9) challenging to solve since it is a joint classification, channel learning and node localization problem. To tackle this difficulty, we split (7.9) into two sub-problems of learning the channel and localizing nodes. We also leverage the 3D map of the city for the measurements classification which will be discussed next.

Radio Channel Learning

Our aim is to learn the radio channel using collected radio measurements from the IoT nodes with known location (anchor nodes). Since the characteristic of the radio channel is independent of the node location and only affected by the structure of the city and the blocking objects in the environment, learning the radio channel from the nodes with known location can provide a good approximation. The measurements are classified by leveraging the 3D map of the city, since for a node with known location the classification variables $\omega_{t,k}$ can be directly inferred. Having classified the measurements, we use a neural network with parameters $\boldsymbol{\vartheta}$ as an approximation of $\psi_{\boldsymbol{\vartheta}}(.)$. The neural network accepts an input vector $[d_{t,k}, \phi_{t,k}, w_{t,k}]^{\mathrm{T}}$ and returns an estimate of the channel gain $\hat{g}_{t,k}$. Just by considering the anchor nodes, problem (7.9) can be rewritten as follows

$$\min_{\substack{\boldsymbol{\vartheta} \\ k\in\mathcal{U}_{\text{known}},\forall t}} \quad \mathcal{L}. \tag{7.10}$$

This optimization is a standard problem in machine learning and can be solved using any gradient-based optimizer. The parameters obtained by solving (7.10) are denoted by $\boldsymbol{\vartheta}^*$.

Node Localization

Having learned the radio channel, we continue to localize the unknown nodes. The optimization problem (7.9) for the set of unknown nodes and utilizing the learned radio channel can be reformulated as follows:

$$\min_{\substack{\omega_{t,k},\,\mathbf{u}_k,\forall t \\ k\in\mathcal{U}_{\text{unknown}}}} \quad \mathcal{L}^* \tag{7.11a}$$

$$\text{s.t.} \quad \omega_{t,k} \in \{0,1\}, k \in \mathcal{U}_{\text{unknown}}, \forall t, \tag{7.11b}$$

where $\mathcal{L}^*$ is obtained by substituting the learned channel model $\psi_{\boldsymbol{\vartheta}^*}(.)$ in (7.8). The binary random variables $\omega_{t,k}$, and the non-linear and non-convex objective function $\mathcal{L}^*$ make problem (7.11) hard to solve. We use the PSO algorithm which is suitable for solving various non-convex and non-linear optimization problems. PSO is a population-based optimization technique that tries to find the solution to an optimization problem by iteratively trying to improve a candidate solution with regard to a given measure of quality (or objective function). The algorithm is initialized with a population of random solutions, called particles, and a search for the optimal solution is performed by iteratively updating each particle's velocity and position based on a simple mathematical formula (for more details on PSO see [146]).

For ease of exposition, we first solve (7.11) by assuming only one unknown node. Then we will generalize our proposed solution to the multi-node case. To apply the PSO algorithm, we define each particle to have the following form

$$\mathbf{c}_j = [x_j, y_j]^{\mathrm{T}} \in \mathbb{R}^2, j \in [1, C], \tag{7.12}$$

where C is the number of particles and each particle represents a possible node location in the city. Therefore, the negative log-likelihood (7.8) for a given particle can be rewritten

as follows

$$\mathcal{L}^*(\mathbf{c}_j^{(i)}) = \log\left(\frac{\sigma^2_{\text{LoS}}}{\sigma^2_{\text{NLoS}}}\right)|\mathcal{M}_{\text{LoS},1,j}| + \sum_{z\in\{\text{LoS},\text{NLoS}\}}\sum_{t\in\mathcal{M}_{z,1,j}}\frac{1}{\sigma_z^2}\left|g_{t,1}-\psi_{\boldsymbol{\vartheta}^*}(d_{t,k},\phi_{t,k},z)\right|^2, \tag{7.13}$$

where $\mathbf{c}_j^{(i)}$ is the j-th particle at the i-th iteration of the PSO algorithm, and $\mathcal{M}_{z,1,j}$ is a set of time indices of measurements collected from node 1 which are in segment z by assuming that the location of node 1 is the same as particle j. To form $\mathcal{M}_{z,1,j}$, a 3D map of the city is utilized. For example, measurement $g_{n,1}$ is considered LoS, if the straight line passing through $\mathbf{c}_j^{(i)}$ and the drone position $\mathbf{p}(t)$ lies higher than any buildings in between. Therefore, the best particle minimizing (7.13) can be obtained from solving the optimization

$$j^* := \arg\min_{j\in[1,C]} \mathcal{L}^*(\mathbf{c}_j^{(i)}), \tag{7.14}$$

where j^* is the index of the best particle which minimizes the objective function in (7.14). In the next iteration of the PSO algorithm, the position and the velocity of particles are updated and the algorithm repeats for τ iterations. The best particle position in the last iteration is considered as the estimate of the node location.

Note that for the multi-node case, without loss of optimality, the problem can be transformed to the multi single-node localization problem and then each problem can be solved individually. This is a consequence of the fact that the radio channel is learned beforehand and is assumed to have the same characteristics for all UAV-node links.

7.3.2 Algorithm

The proposed Algorithm 2 iterates between three phases: 1) the agent uses a policy obtained from the DQN (initially random) to collect real-world RSS measurements from the IoT nodes; 2) the collected measurements are used to learn the radio channel and localize the nodes with unknown locations as described in sections 7.3.1 and 7.3.1; 3) the agent performs a new set of experiments in the simulated environment under the learned radio channel model using the estimate of the node locations to train the DQN. Then, the agent repeats the first phase of the algorithm by generating a new policy using the trained Q-network and the procedure continues until convergence.

The experience replay buffers for real world and simulated world experiments are denoted $\mathcal{D}$ and $\tilde{\mathcal{D}}$, respectively. A new episode in phase 1 and phase 3 starts by resetting the time index, the initial UAV position and the battery budget (lines 7 and 22). To train the DQN, an ϵ-greedy exploration technique is used (line 36) with decay constant κ (2.15). η is the learning rate for primary network parameters θ. Target network parameters are updated every N_{target} episodes with a hard update. In phase 3, the algorithm performs I sets of experiments in the simulated world, and the whole algorithm terminates after carrying out E_{max} real-world experiments.

Algorithm 2 Model-aided DQN trajectory design

Initialize replay buffer $\mathcal{D}, \tilde{\mathcal{D}}$
Initialize primary network parameters θ and target network parameters $\bar{\theta}$
Initialize episode counter for real-world and simulated training episodes $n = 0$
for $e = 0$ to E_{max} **do**
  $n = n + 1$
  1) Real-world experiment:
  Initialize $s_0 = (\mathbf{p}(\mathbf{t}) = \mathbf{v}_\mathrm{I}, b_{max})$, $t = 0$
  while $b(t) \geq 0$ **do**
    $a_t = \arg\max_{a'} Q_\theta(s_t, a')$
    Validate a_t using the safety controller (7.5)
    Observe $r_t, s_{t+1}, \gamma_{1,t}, \cdots, \gamma_{K,t}$
    Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Memorize $(\mathbf{p}_t, \gamma_{1,t}, \cdots, \gamma_{K,t})$
    $t = t + 1$
  end while
  2) Learning the environment:
  Learn the radio channel as described in Section 7.3.1
  Localize unknown nodes as described in Section 7.3.1
  3) Simulated-world experiment:
  for $i = 0$ to I **do**
    $n = n + 1$
    Initialize $\tilde{s}_0 = (\mathbf{p}(t) = \mathbf{v}_\mathrm{I}, b_{max})$, $t = 0$
    while $b(t) \geq 0$ **do**
      $\tilde{a}_t = \begin{cases} \text{randomly select from } \mathcal{A} & \text{with probability } \epsilon \\ \arg\max_{a'} Q_{\bar{\theta}}(\tilde{s}_t, a') & \text{otherwise (2.15)} \end{cases}$
      Validate $\tilde{a}_t$ using the safety controller (7.5)
      Compute $\tilde{r}_t$ from (7.3), and $\tilde{s}_{t+1}$ from (7.1), (7.2)
      store $(\tilde{s}_t, \tilde{a}_t, \tilde{r}_t, \tilde{s}_{t+1})$ in $\tilde{\mathcal{D}}$
      for $i = 0$ to m **do**
        Sample (s_i, a_i, r_i, s_{i+1}) uniformly from $\{\mathcal{D} \cup \tilde{\mathcal{D}}\}$
        $y_i = \begin{cases} r_i & \text{if terminal} \\ r_i + \gamma \max_{a'} Q_{\bar{\theta}}(s_{i+1}, a') & \text{otherwise according to (2.13)} \end{cases}$
        Compute $L_i^{\mathrm{DQN}}(\theta) = \mathbb{E}\left[\left(Q_\theta(s_i, a_i) - Y_i^{\mathrm{DQN}}\right)^2\right]$ according to (2.12)
      end for
      $\theta \leftarrow \theta - \eta \cdot \frac{1}{m} \nabla_\theta \sum_{i=1}^{m} L_i^{\mathrm{DQN}}(\theta)$
      $t = t + 1$
    end while
    $\epsilon \leftarrow \epsilon_{final} + (\epsilon_{start} - \epsilon_{final}) e^{-\kappa n}$
    if $(n \mod N_{target} = 0)$ **then** $\bar{\theta} = \theta$
  end for
end for

7.4 Simulations

We consider a dense urban city neighborhood comprising buildings and regular streets as shown in Fig. 7.1. The height of the buildings is Rayleigh distributed in the range of 5 to 40 m and the true propagation parameters are identical to chapter 4. The UAV collects radio measurements from the ground nodes every 5 m and we assume that the altitude of the UAV is fixed to 60 m during the course of its trajectory, above the height of all buildings. The mission time of each episode is fixed to $T = 20$ time steps with a fixed UAV movement step size of $c = 50$ m.

We assume there are six ground nodes. Only the locations of anchor nodes $\mathbf{u}_1$ and $\mathbf{u}_2$ are known to the UAV in advance. The UAV starts from $\mathbf{v}_\mathrm{I} = [100, 100, 60]^\mathrm{T}$ and needs to reach the destination point $\mathbf{v}_\mathrm{F} = [300, 400, 60]^\mathrm{T}$ by the end of the mission. To learn the channel, we use a NN with two hidden layers where the first layer has 60 neurons with tanh activation function, and the second layer 30 neurons with ReLU activation function. The DQN comprises 2 hidden layers each with 120 neurons and ReLU activation function.

In Fig. 7.3, we compare the performance of the baseline Q-learning algorithm as explained in section 2.3.3 and akin to chapter 3, with the proposed model-aided Q-learning algorithm. Moreover, we show the result of an algorithm similar to chapter 4, where the mixed-radio map of the nodes is embedded in the state vector. Fig. 7.2 shows a plot of the normalized combined radio map for the scenario in Fig. 7.1. In reference to chapter 4, where we showed that centering the map on the UAV position is highly beneficial for learning performance, we feed the centered map from Fig. 7.2b to the agent. To compute the mixed-radio map, the individual radio maps of all nodes are combined. Individual radio maps are computed using the 3D map of the city and assuming perfect knowledge node positions and the radio channel. The model-aided algorithm outperforms the other approaches since it merely requires 10 real-world experiments episodes to converge to the same performance level as other algorithms. The radio map-based algorithm is superior to the baseline since it uses more information in a more efficient form, i.e. the map and perfect knowledge of node positions and the radio channel model.

Fig. 7.1 shows the final trajectory after convergence of algorithm 2. The UAV starts flying towards the closest node and hovers above for several time steps in order to maximize the amount of collected data, and then reaches the destination $\mathbf{v}_\mathrm{F}$. Moreover, the estimate of unknown node locations obtained at the last episode of the training phase of Algorithm 2 are shown and confirmed to be very close the true positions.

7.5 Conclusion

We have introduced a novel model-accelerated DRL path planning algorithm for UAV data collection from distributed IoT nodes with only partial knowledge of the nodes' locations. In comparison to two standard deep Q-learning algorithms, using either full or no knowledge of sensor node locations, we have demonstrated that the model-aided approach requires at least one order of magnitude less training data samples to reach the same data collection performance. However, this was demonstrated for a single mission scenario at a time, in contrast to chapters 4, 5, and 6, where we learned a control

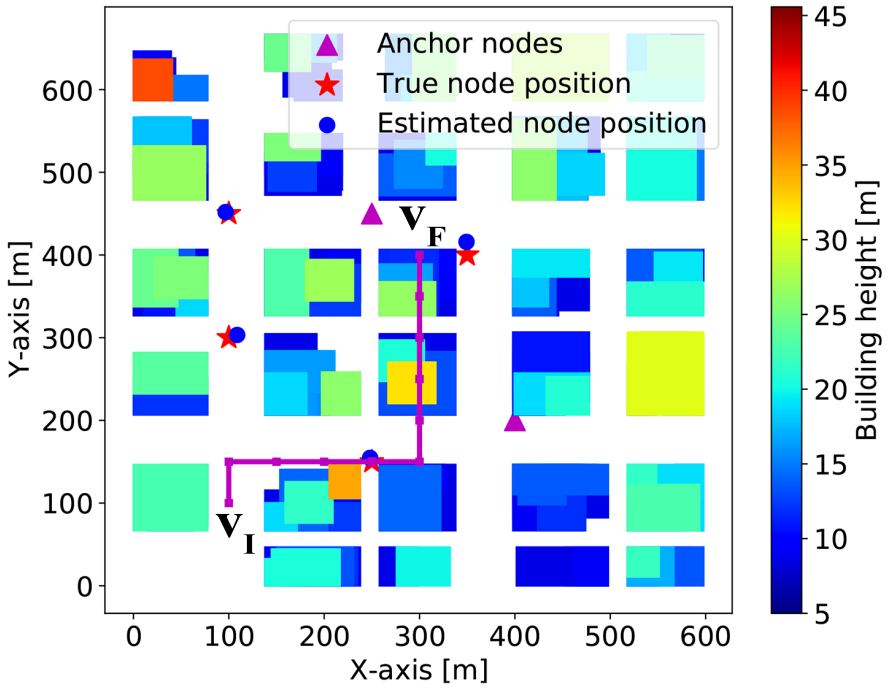
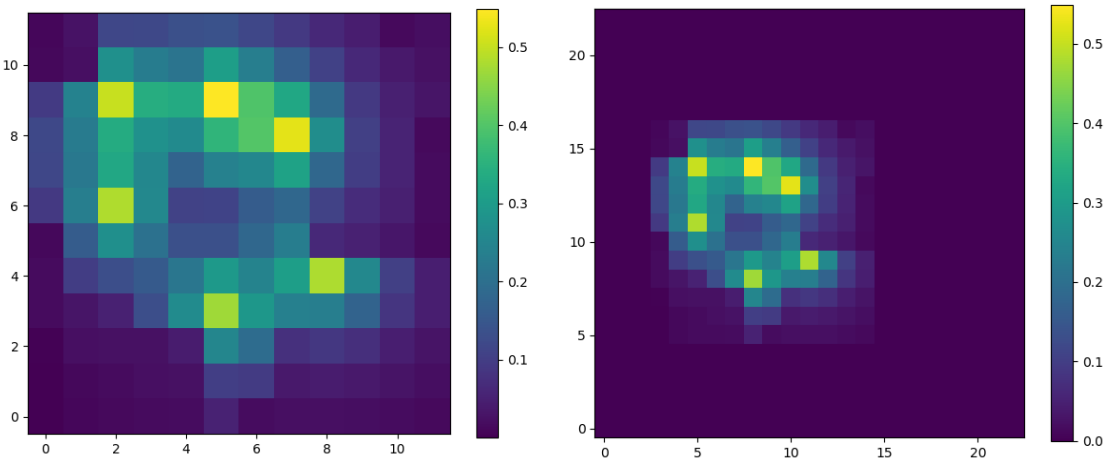


Figure 7.1 – Trajectory obtained by model-aided Q-learning and the estimates of unknown node locations in the final episode of Algorithm 1.

policy that generalizes on mission parameters, which is a more complex problem. In summary, the model-aided approach is a promising step in the direction of real-world UAV trajectory planning with deep reinforcement learning. In future work, we also plan to extend this method to the multi-UAV case in CTDE and/or fully decentralized settings.



(a) Non-centered radio map.

(b) Centered radio map with padding.

Figure 7.2 – The normalized, combined radio map for the scenario in Fig. 7.1 with six users, before and after map centering.

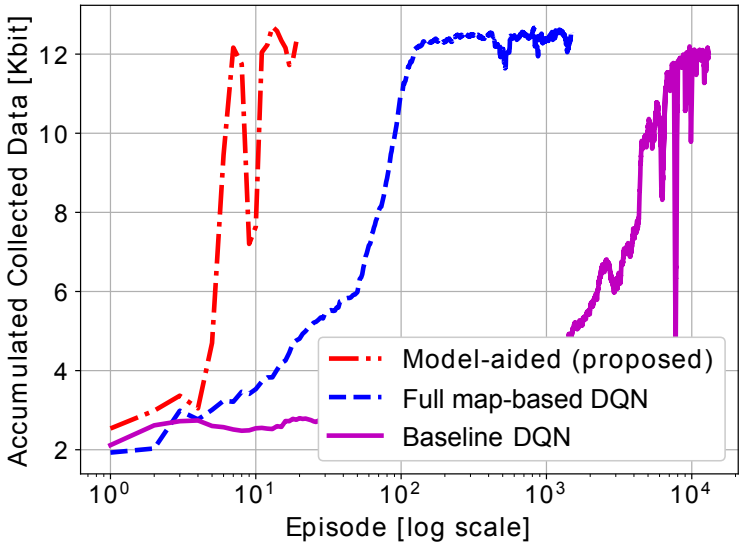


Figure 7.3 – Comparison of proposed model-aided, full-knowledge map-based (chapter 4), and no prior knowledge baseline DQN (chapter 3), showing accumulated collected data versus training episodes on a logarithmic scale.

Chapter 8

Conclusion

In this dissertation, we have focused on the design of trajectories for drones in UAV-aided wireless networks, with the UAVs' mobility as the most central difference and advantage compared to classical stationary network infrastructure. Reinforcement learning as a method of trajectory design thereby mirrors the underlying challenge directly: an autonomous UAV agent needs to make instantaneous control decisions that balance the system's communication goals with other constraints such as limited flying time, obstacle avoidance or multi-UAV coordination with varying degrees of prior information available about the environment and mission to support its decisions.

We have started this investigation with a scenario where virtually no prior information is available to an aerial base station serving ground users before starting its mission. We showed that the RL paradigm, through exploration of the state space and then exploitation of learned knowledge, allows the UAV agent to converge to the optimal control policy while efficiently integrating landing spots into the trajectory that allow the aerial BS to extend its active mission time. The training procedure was shown to adapt to complex and random environmental effects, albeit at the price of long training time.

In the following parts of this thesis, we have offered two approaches to mitigate this drawback. First, by learning a generalized control policy that is able to adapt instantly to changes in the defining parameters of the UAV mission, we have shown that it is possible to take away the need for lengthy retraining when the scenario changes, including when the number of UAVs that are deployed in a team changes. Secondly, we have offered an approach for model-aided acceleration of the training process that minimizes the need for expensive training data collection in the real world. Both approaches were demonstrated to work effectively in a scenario where UAV data harvesters are deployed in a dense urban environment to collect data from distributed Internet of Things sensor nodes. Here, it was assumed that a map of the city is available for the UAV agents to make movement decisions.

We have shown that it is of high importance how the additional knowledge available to the agents is exploited. By centering the map on the agent's position and feeding it into convolutional network layers, training performance is increased considerably. We have also introduced a dual map processing approach that divides the available information into a local and a global environment map, which allows the approach to scale to even

larger and more complex environments. Furthermore, by describing the UAV trajectory planning problem based on map layers, we have shown that the proposed DRL methods can be directly used to solve other instances of UAV trajectory planning, such as coverage path planning.

In this thesis, we have laid some ground work and suggested steps towards the goal of deploying DRL trajectory planning to real-world UAV-aided communication networks. Although DRL has shown great promise in many application areas, actual non-prototype use in the real-world is still rare due to a number of high-level challenges [75], one of the most important ones is the expensive training data collection process. In addition to the concepts presented in this thesis, investigations in future work can explore the combination with other existing approaches such as transfer learning [147], where the goal is to improve learning performance by transferring knowledge contained from different but related source domains. Multi-task reinforcement learning [148] is interesting for cases where the UAV needs to perform multiple tasks in the same mission scenario, where the aim is to leverage the learned knowledge contained in multiple related tasks to help improve the generalization performance for all tasks. A very promising research direction is also the concept of training RL agents from offline data, i.e. training data that was collected previously and that can be used without repeated direct interaction with the real-world environment [149]. Apart from the combination with new concepts, future improvements of the approaches presented in this paper could include the extension of the UAV action space to continuous 3D control, the investigation of attention-based mechanisms for map processing and allowing for adaption to even larger maps with macro-actions or options [150].

Appendices

Annexe A

Résumé

A.1 Introduction

La technologie et les applications potentielles des drones, aussi communément appelés par l'acronyme anglais UAV, ont connu une innovation rapide au cours de la dernière décennie. Des avancées cruciales en matière de matériel, de fabrication et de coût des UAV [1], associées à des cadres réglementaires nouvellement créés pour l'utilisation commerciale des UAV [4], ont conduit à la création d'un marché qui devrait atteindre 63,6 milliards de dollars d'ici à 2025 [5].

Un scénario d'application fréquemment cité est la livraison de marchandises aux consommateurs sur le dernier kilomètre par des drones. Un projet commercial réussi de drones autonomes livrant de la nourriture et des produits d'épicerie aux clients de Reykjavik, la capitale de l'Islande, a été lancé dès 2018, plaçant les autorités aériennes islandaises en tête de liste des pays ayant autorisé une certaine forme de vol de drone autonome [7]. Un exemple dans le contexte du maintien en état des infrastructures vieillissantes du monde est la société japonaise Hitachi, qui déploie déjà commercialement des drones partiellement autonomes qui collectent des données de maintenance à partir de capteurs de l'Internet des objets (IoT) intégrés dans de grandes structures, comme les ponts de San Juanico et d'Agas-Agas aux Philippines [8]. Les drones peuvent même être utilisés pour sauver des vies : leur déploiement flexible et rapide les rend très intéressants pour les scénarios où les infrastructures fixes (par exemple, les réseaux de communication) sont détruites ou indisponibles, c'est-à-dire dans les situations de catastrophe et de recherche et sauvetage. Cela peut être le cas lorsqu'une catastrophe naturelle, comme un tremblement de terre, détruit les stations de base (BS) et que les drones font office de stations de base volantes pour rétablir les capacités de communication des premiers intervenants [9]. Pour réaliser leur potentiel en faveur de la société du futur, il est clair que tous les drones ont besoin de capacités de communication bien conçues ou fournissent eux-mêmes des services de communication.

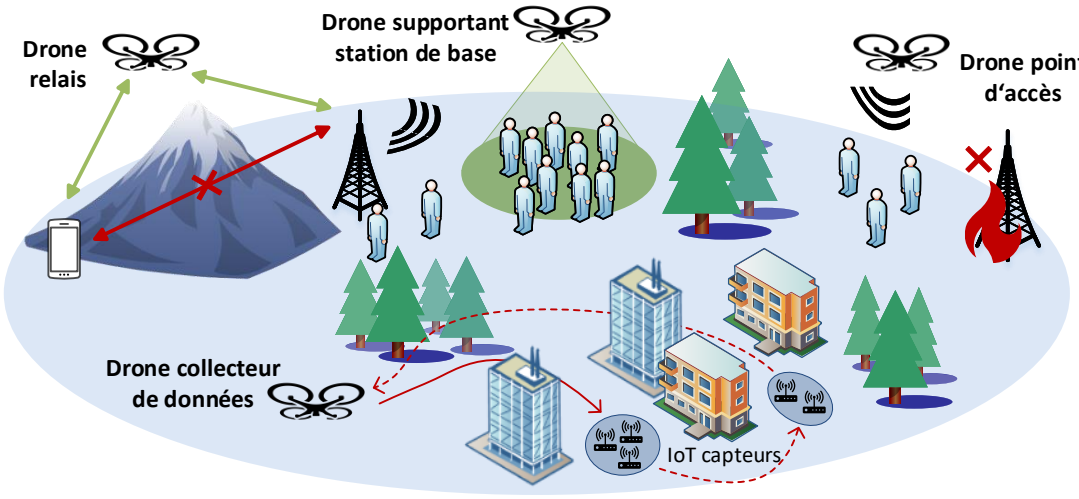


FIGURE A.1 – Exemples d'applications pour les drones fournissant des services de communication et soutenant l'infrastructure stationnaire.

A.1.1 Réseaux assistés par drones

Ces dernières années, l'intérêt pour l'intégration des drones dans les réseaux de communication sans-fil n'a cessé de croître, tant dans la sphère commerciale que dans celle de la recherche universitaire. Fondamentalement, les drones dans les réseaux de communication peuvent être envisagés de deux manières et décrits comme [1, 10] : *drones connectés par voie cellulaire* qui sont attachés aux réseaux mobiles avec un lien de commande et de contrôle en tant que terminaux de réseau ou équipement utilisateur aérien (UE), et les fournisseurs de services de communication dans les *réseaux de communication assistés par drone*, également appelés réseau d'accès radio volant (FRAN). Cette thèse se concentre sur les réseaux de communication assistés par drones.

Les progrès réalisés dans le matériel des drones ainsi que la miniaturisation des équipements de communication sans fil ont permis une multitude d'applications potentielles pour les réseaux sans fil assistés par drones, dont certaines sont représentées sur la figure A.1. Un cas d'utilisation naturel consiste à attacher un point d'accès (AP) sans fil comme charge utile à un drone qui peut alors fournir une capacité de communication supplémentaire dans les zones où les réseaux terrestres sont encombrés [27, 28]. Un autre cas d'utilisation de ces stations de base aériennes est l'établissement de services de communication autonomes, par exemple dans des zones qui ne disposent pas de couverture ou dans lesquelles l'infrastructure terrestre est désactivée [29, 30]. Les relais de drones [31, 32] sont susceptibles de pouvoir établir des liaisons en visibilité directe (LoS) vers des utilisateurs obstrués, grâce à leur haut degré de mobilité et d'altitude. Les réseaux de capteurs de l'Internet des objets (IoT) soumis à des contraintes d'énergie et de débit peuvent être pris en charge par des collecteurs de données de drones capables de décrire un modèle de vol

qui les amène à proximité des dispositifs, augmentant ainsi l'efficacité énergétique [33–35].

Par rapport aux réseaux terrestres fixes, la communication assistée par drone présente un certain nombre de défis uniques en matière de conception et de recherche [36]. Tout d'abord, le déploiement des stations de base aériennes dans un espace 3D offre un degré de liberté (DoF) supplémentaire par rapport au déploiement typiquement 2D des stations de base terrestres. Un autre défi majeur est l'environnement de propagation différent : [37]. Le canal air-sol (AG) pour les drones n'a pas été étudié de manière trop détaillée auparavant, pourtant des modèles de propagation précis sont essentiels pour développer des protocoles et des techniques de communication robustes. Enfin, les stations de base et les relais aériens ajoutent des contraintes supplémentaires à la maximisation des performances de communication ou de la qualité de service (QoS), telles que le temps de vol limité et les exigences intrinsèques de la dynamique de vol. Les protocoles d'ordonnancement, d'allocation de recherche et d'accès multiple doivent être adaptés pour tenir compte de ces changements.

Ces défis guident la recherche sur les communications assistées par drone. La mobilité peut être prise en compte soit par l'optimisation du placement des stations de base/relais des drones, soit par l'optimisation complète de la trajectoire de bout en bout. Même le placement des drones est un problème plus difficile que celui des stations de base stationnaires, en raison de la profondeur de champ supplémentaire. Les algorithmes pour le placement des stations de base UAV vont de la force brute [38], la programmation génétique [27], le regroupement K-means [25] à la théorie des contrats et l'apprentissage automatique (ML) [39]. L'optimisation de la trajectoire complète a été abordée, par exemple par l'optimisation convexe séquentielle [40], l'optimisation fonctionnelle et le contrôle optimal [41], ou l'apprentissage par renforcement (RL) [42]. L'amélioration de la modélisation des canaux AG nécessite des études expérimentales telles que [14, 43]. Des enquêtes et des tutoriels qui résument les recherches récentes sont donnés par [1, 10, 44].

A.1.2 Apprentissage automatique pour les communications par drones

L'expansion des applications potentielles des drones et les progrès technologiques dans le matériel des drones ont également conduit à l'émergence de nouveaux défis et problèmes associés. Les solutions à ces défis basées sur l'intelligence artificielle (AI) et l'apprentissage machine (ML) sont devenues un domaine de recherche très actif. Une croissance explosive de l'intérêt pour les deux aspects de cette thèse, les communications assistées par drone et le ML, a coïncidé au cours des dernières années, conduisant à une multitude de résultats de recherche qui combinent les deux.

Le cadre de l'apprentissage par renforcement est particulièrement adapté aux défis posés par le déploiement de drones autonomes dans les réseaux de communication, car l'idée centrale est celle d'un agent autonome prenant des décisions (par exemple, la planification d'une trajectoire) pour maximiser un certain objectif (par exemple, la qualité de service pour une station de base aérienne) dans un environnement inconnu. L'apprentissage par renforcement profond (DRL) peut être utilisé dans de nombreux cas de placement et de planification de trajectoire de drones, comme la planification de trajectoire multi-drones sensible aux interférences [18], la collecte de données dans le

contexte de la détection de foule mobile [66], ou la maximisation de la couverture des communications dans les réseaux assistés par drones [30]. L'allocation des ressources est l'une des applications des réseaux de drones qui vont au-delà de la planification des chemins. Dans [67], l'apprentissage par renforcement multi-agent (MARL) est utilisé pour sélectionner automatiquement l'utilisateur communicant, le niveau de puissance et le sous-canal de chaque drone sans aucun échange d'informations entre les drones. Un algorithme d'association d'utilisateurs basé sur le RL est développé dans [68] pour minimiser les transferts d'utilisateurs dans un réseau de drones BS. La combinaison de drones avec des surfaces réfléchissantes intelligentes dans le contexte de la 6G donne également lieu à divers défis qui peuvent être résolus par RL [26].

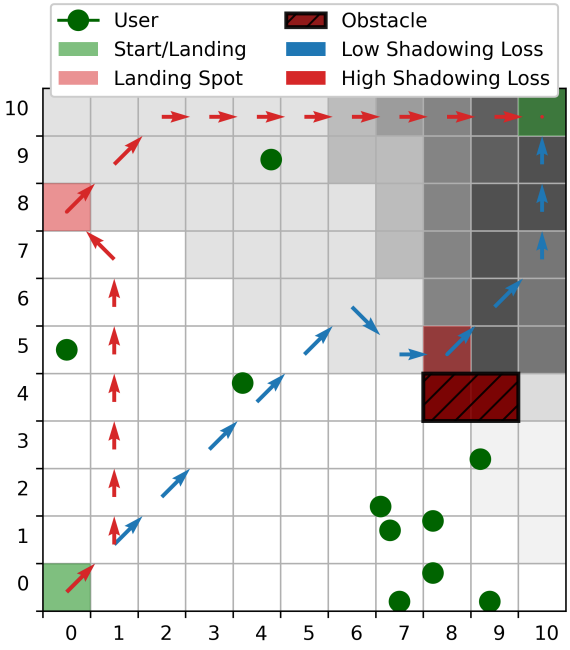
Si les approches basées sur le ML se sont révélées très prometteuses dans de nombreuses applications de communication par drone, il est important de souligner que cela ne signifie en aucun cas que le ML surpasse généralement les approches classiques dans tous les scénarios et applications. En fait, le ML apporte ses propres défis par rapport aux approches classiques : il sacrifie l'interprétabilité et les performances garanties dans certains scénarios, tandis que les gains de performances dans le monde réel ne se sont pas matérialisés dans de nombreux domaines, en raison du fait que le déploiement pratique dans le monde réel des solutions basées sur le ML est souvent encore difficile à réaliser. Cependant, il est également vrai que le ML offre la possibilité de trouver des solutions viables à des problèmes qui sont difficiles ou impossibles à aborder avec des approches classiques.

Une enquête exhaustive sur les thèmes de recherche actuels en AI pour les réseaux sans fil basés sur les drones est donnée par [61]. Bithas et al. [71] ont fourni une vue d'ensemble avec une sélection similaire de sujets, tandis que [52] se concentre sur les approches DRL.

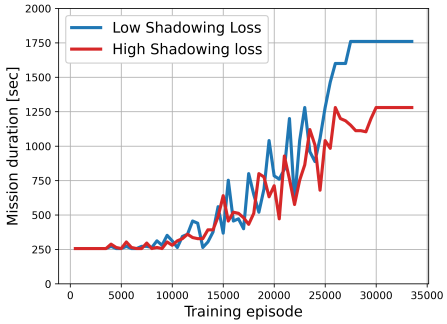
A.2 Planification de trajectoire pour station de base aérienne avec points d'atterrissage (Chapitre 3)

Comme ce travail se concentre sur les petits drones polyvalents de type multirotor, dont le quadcopter est l'exemple le plus couramment utilisé, les restrictions en matière de consommation d'énergie et de densité d'énergie de la batterie sont des contraintes centrales pour la planification des missions de BS aérienne. Même en transformant un quadrirotor en "batterie volante", la durée maximale de la mission ne peut généralement pas dépasser deux heures [97]. Comme la consommation d'énergie pour le vol dépasse généralement de loin la consommation d'énergie de la station de base transportée, le concept de points d'atterrissage (LS) a été introduit dans [95] pour prolonger la durée de la mission. Un point d'atterrissage est une petite parcelle d'immobilier, par exemple un toit, où une station de base de drone peut se poser, ce qui permet d'économiser de l'énergie tout en continuant à servir les utilisateurs de service mobile. Souvent, l'utilisation du LS s'accompagne du sacrifice d'une certaine qualité de service instantanée pour certains utilisateurs, mais permet d'allonger la durée globale de la mission.

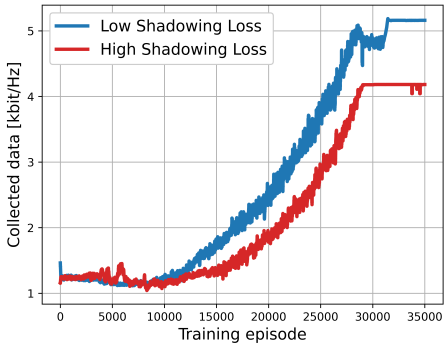
Nous démontrons dans ce chapitre l'utilisation d'un réseau Q profond (DQN), une



(a) Trajectoire finale du drone BS.



(b) Durée de la mission sur les épisodes de formation.

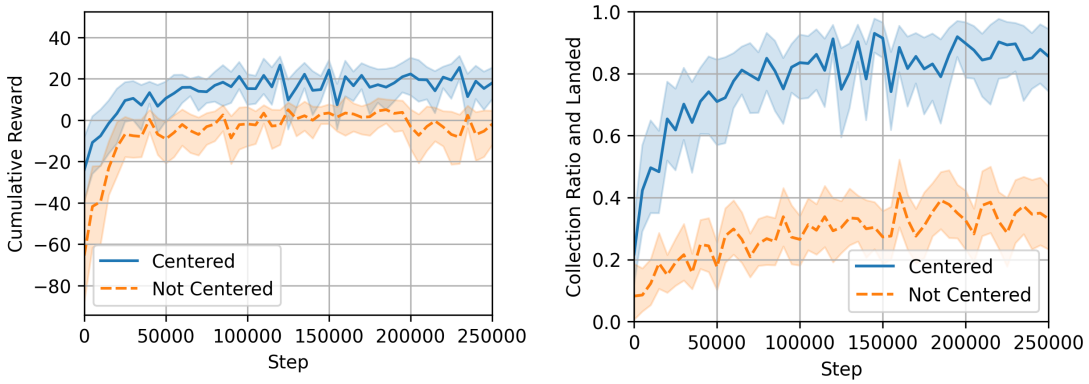


(c) Données collectées au cours d'épisodes de formation.

FIGURE A.2 – Scénario comportant $L = 2$ points d'atterrissage et un obstacle qui provoque l'ombrage de certains utilisateurs sur la carte. Les résultats finaux pour une perte d'ombre élevée et faible sont comparés.

technique d'apprentissage par renforcement, pour planifier la trajectoire d'un UAV BS tout en servant un groupe d'utilisateurs au sol en maximisant la somme du taux d'information sur toute la durée du vol avec une quantité limitée d'énergie dans la batterie du drone au départ. Les décisions de mouvement sont donc prises en fonction de la position actuelle du drone et du contenu de la batterie, ainsi que de l'espérance du taux de somme totale qui peut être atteint jusqu'à ce que la batterie soit épuisée. Pour économiser de l'énergie pendant la mission, le drone est autorisé à atterrir dans les LS désignés.

Il est important de noter que, contrairement aux scénarios décrits dans les chapitres suivants de cette thèse, le drone n'a absolument aucune connaissance préalable de l'environnement ou du scénario de la mission. Cela signifie qu'il n'a aucune connaissance de l'existence des LS, du nombre et de l'emplacement des utilisateurs, des obstacles, du modèle de canal ou de l'emplacement de la zone d'atterrissage. En référence au dilemme exploration-exploitation, cela signifie que le drone doit découvrir toutes les caractéristiques de l'environnement par essais et erreurs. Nous comparons également les performances de collecte de données du système DQN avec la trajectoire optimale qui



(a) Récompense cumulative épisodique.

(b) Taux de recouvrement et atterrissage réussi.

FIGURE A.3 – Comparaison des processus de formation entre les cartes centrées et non centrées, montrant la moyenne et les quantiles à 99 % de trois processus de formation chacun, avec des métriques épisodiques regroupées dans des bins de 5000.

peut être calculée à l'aide de la programmation dynamique (DP) basée sur une approche similaire à celle de [95]. En revanche, l'approche DP nécessite bien sûr une connaissance préalable complète de l'environnement et du modèle.

La figure A.2 présente un exemple de mission de BS de drone dans un environnement comportant un obstacle et deux points d'atterrissage. L'adaptation des drones à des conditions de perte d'ombrage élevées et faibles est illustrée.

A.3 Collecte de données par drone dans les réseaux IoT multi-scénario (Chapitre 4)

Dans le chapitre précédent, nous avons montré qu'une approche de planification de trajectoire basée sur DQN pour les stations de base aériennes permet de concevoir des trajectoires qui atteignent des niveaux de performance optimaux - sans aucune hypothèse sur le modèle sous-jacent et sans aucune connaissance préalable de l'environnement ou du scénario. Cela va de pair avec une grande demande de données d'entraînement, car l'agent du drone doit déduire toutes les informations de l'exploration par essais et erreurs. Plus problématique encore, si un élément du scénario change, comme la position de l'utilisateur, la procédure d'apprentissage doit être relancée afin de s'adapter au nouveau scénario. Cependant, dans de nombreux cas d'application, certaines informations supplémentaires sur l'environnement sont disponibles, par exemple une carte de l'environnement. Dans ce chapitre, nous souhaitons utiliser les informations cartographiques pour concevoir un algorithme DRL capable de généraliser la conception de trajectoire sur plusieurs scénarios, c'est-à-dire de prendre en compte les paramètres variables du scénario pendant l'apprentissage et de trouver une politique de planification de trajectoire qui s'adapte instantanément à un changement de scénario et ne nécessite pas de réapprentissage, ce qui est un problème beaucoup plus complexe. Si les méthodes RL profondes doivent être

appliquées dans des missions réelles, la demande prohibitive de données d'entraînement pose l'un des défis les plus sévères [75]. En prenant en compte des paramètres variables dans la conception et la formation du modèle de réseau neuronal, nous faisons un pas vers l'atténuation de ce défi.

Nous passons également du scénario d'une station de base aérienne à un drone en mission de collecte de données à partir de dispositifs distribués de l'Internet des objets (IoT). Par exemple, les opérateurs de l'IoT peuvent déployer des collecteurs de données par drone en l'absence d'infrastructure cellulaire coûteuse à proximité. Une autre raison est le bénéfice de l'efficacité du débit lié à la présence de drones qui décrivent un modèle de vol qui les rapproche des dispositifs IoT.

La collecte de données à partir de dispositifs de capteurs dans un environnement urbain impose des contraintes difficiles sur la conception de la trajectoire des drones autonomes. La densité énergétique de la batterie limite fortement la durée de la mission pour les drones quadrirotors, tandis que l'environnement urbain complexe pose des problèmes d'évitement des obstacles et de respect des zones d'interdiction de vol (NFZ) réglementaires. De plus, le canal de communication sans fil est caractérisé par de fréquentes fluctuations de l'atténuation par l'alternance de liaisons en visibilité directe (LoS) et de liaisons sans visibilité directe (NLoS). Les approches DRL offrent la possibilité d'équilibrer les défis et l'objectif de collecte de données pour les environnements complexes d'une manière simple en les combinant dans la fonction de récompense. Une autre raison de la popularité du paradigme DRL dans ce contexte est l'efficacité computationnelle de l'inférence DRL. DRL est également l'une des rares méthodes qui nous permet de nous attaquer directement à la tâche complexe.

Enfin, nous avons également montré dans ce chapitre qu'il y a une augmentation considérable de l'efficacité d'apprentissage pour l'agent RL lorsqu'il exploite une carte de l'environnement centrée sur la propre position de l'agent drone. Ceci est visualisé dans la figure A.3. L'avantage d'utiliser une carte centrée est le résultat d'un changement de position auquel correspond un neurone de la couche "Flatten". Si la carte n'est pas centrée, les neurones de cette couche correspondent à des caractéristiques à des positions *absolues*. Si la carte est centrée, ils correspondent à des éléments en position *relative* par rapport à l'agent. Comme les actions de l'agent sont uniquement basées sur sa position relative par rapport aux caractéristiques, par exemple sa distance par rapport aux dispositifs, l'efficacité de l'apprentissage augmente considérablement.

A.4 Coordination de plusieurs drones dans la collecte de données multi-scénario (Chapitre 5)

Dans ce chapitre, nous étendons et améliorons les résultats du précédent tout en considérant un scénario similaire de communications assistées par drone avec collecte de données à partir de dispositifs IoT distribués. Il est raisonnable de supposer qu'un opérateur de réseau déployant des drones de collecte de données dans le cadre de son infrastructure de réseau IoT, tenterait d'augmenter l'efficacité du système en déployant plusieurs de ces drones. Du point de vue de la planification de trajectoire, cela nécessite de prendre

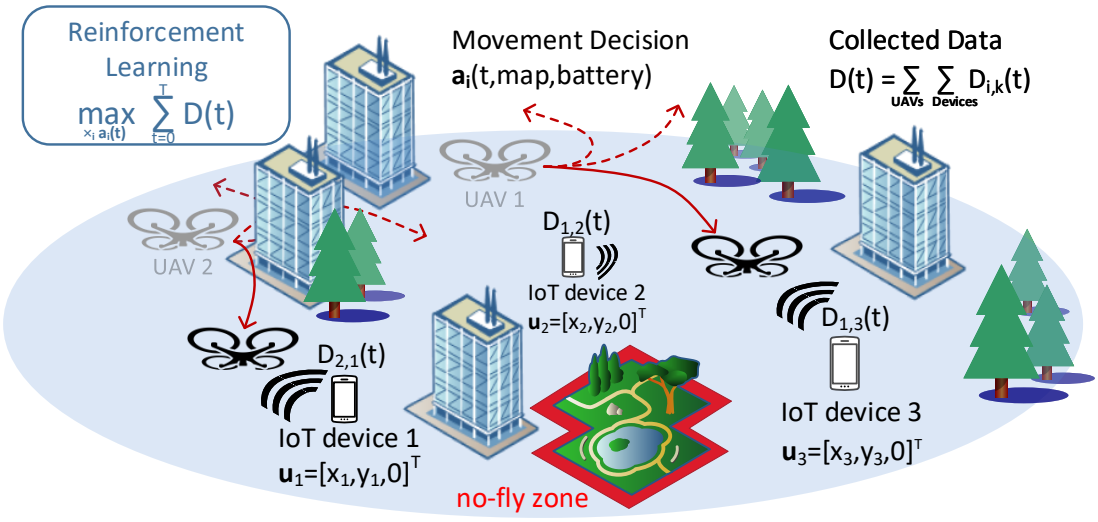


FIGURE A.4 – Vue d'ensemble du scénario de collecte de données par plusieurs drones. Les drones utilisent une carte globale compressée et une carte locale recadrée pour planifier leurs trajectoires.

en compte l'aspect coordination et une reformulation de l'approche de l'apprentissage par renforcement multi-agent (MARL). Par conséquent, nous formulons le problème de planification de trajectoire pour une équipe de drones coopérative, non communicante et homogène chargée de maximiser les données collectées à partir de capteurs IoT distribués, sous réserve de contraintes de temps de vol et d'évitement des collisions, comme le montre la figure A.4. Le problème de planification de trajectoire est traduit en un processus de décision de Markov partiellement observable décentralisé (Dec-POMDP), que nous résolvons par une approche DDQN, en approximant la politique de contrôle optimale du drone sans connaissance préalable des caractéristiques difficiles des canaux sans fil dans les environnements urbains denses.

La deuxième amélioration est liée au fait que l'approche de centrage de la carte telle qu'introduite précédemment, bien que conduisant à des gains de performance d'apprentissage élevés, augmente également le nombre de paramètres de réseau neuronal entraînables, ce qui la rend non viable pour des cartes plus grandes, plus réalistes et plus complexes. Dans ce chapitre, nous modifions également le scénario pour différencier les grands bâtiments qui agissent comme des obstacles à la navigation, ainsi que les bâtiments plus petits qui peuvent être survolés par les drones. Nous abordons le problème des paramètres croissants en exploitant une combinaison de représentations cartographiques globales et locales centrées de l'environnement qui sont introduites dans les couches convolutives des agents. Nous montrons que l'architecture de réseau que nous proposons permet aux agents de coopérer efficacement en divisant soigneusement la tâche de collecte de données entre eux, de s'adapter à de grands environnements complexes, et de prendre des décisions de mouvement qui équilibrent les objectifs de collecte de données, l'efficacité

du temps de vol et les contraintes de navigation.

Enfin, dans ce travail, nous nous concentrons sur le contrôle d'une équipe de drones, composée d'un nombre variable de drones identiques chargés de collecter des quantités variables de données à partir d'un nombre variable de dispositifs de capteurs IoT stationnaires à des emplacements variables dans un environnement urbain. Cela impose des contraintes difficiles sur la conception de trajectoires pour les drones autonomes. L'apprentissage d'une politique de contrôle qui se généralise sur l'espace des paramètres du scénario nous permet d'analyser l'influence des paramètres individuels sur les performances de collecte et de fournir une certaine intuition sur les avantages au niveau du système, comme le montre la Fig. A.5. Un excellent exemple que nous avons trouvé pour la carte 'Manhattan32' est que le déploiement de plusieurs drones de coordination peut compenser le coût de l'équipement supplémentaire (c'est-à-dire les drones supplémentaires) par une réduction substantielle du temps de mission. Par exemple, il faut deux fois plus de temps de vol ($b_0 = 150$) à un seul drone pour achever la mission de collecte de données que deux drones de coordination ($b_0 = 75$) pour la mener à bien.

En nous référant à la Fig. A.5, nous analysons d'abord les performances de l'agent de la Fig. A.5 dans la plage d'entraînement des paramètres du scénario (lignes pleines), puis nous étendons l'analyse aux scénarios hors distribution (lignes pointillées). La Fig. A.5 montre l'influence des paramètres d'un seul scénario sur le ratio moyen de collecte de données avec atterrissage réussi de tous les agents et indique l'augmentation des performances de collecte lorsque davantage de drones sont déployés. Dans le même temps, un plus grand nombre de drones entraîne des exigences accrues en matière d'évitement des collisions, comme nous l'avons observé à travers un plus grand nombre d'activations de contrôleurs de sécurité dans les premières phases de formation. Comme les capteurs IoT sont positionnés de manière aléatoire dans l'espace cartographique inoccupé, une augmentation du nombre de capteurs entraîne des exigences de trajectoire plus complexes et une baisse des performances, comme le montre la figure A.5b.

La Fig. A.5c montre l'influence de l'augmentation du volume initial de données par capteur sur les performances globales de collecte. Il apparaît que des volumes de données initiaux plus élevés par capteur sont bénéfiques à peu près jusqu'au point de $D_{k,init} \in [10, 12.5]$ unités de données, après quoi les contraintes de temps de vol obligent les drones à abandonner une partie des données, et le rapport de collecte montre une tendance légèrement négative. Une augmentation du temps de vol disponible est clairement bénéfique pour les performances de collecte, comme l'indique la figure A.5d. Cependant, l'effet devient plus faible lorsque la plupart des données sont collectées, et les drones commencent à donner la priorité à la minimisation du temps de vol total et à l'atterrissage en toute sécurité plutôt qu'à la collecte des derniers bits de données.

La Fig. A.5 montre en outre comment les agents réagissent aux paramètres du scénario qui n'ont pas été rencontrés pendant la formation. Les valeurs correspondantes sont mises en évidence par des lignes pointillées. On constate que la performance des agents suit la même tendance que dans le reste des données, lorsqu'on augmente le nombre de capteur (Fig. A.5b) ou les données initiales par capteur (Fig. A.5c) hors de la région entraînée. En augmentant le temps de vol maximum (Fig. A.5d) pour les agents Manhattan32, ou en le diminuant pour les agents Urban50, le taux de collecte avec des performances

d'atterrissage réussi, augmente ou diminue en conséquence. L'augmentation du nombre d'agents à quatre (Fig. A.5a) réduit légèrement les performances. La raison en est la diminution des performances d'atterrissage. Cependant, il faut s'y attendre puisque la probabilité que tous les agents atterrissent diminue avec le nombre d'agents. Comme le ratio de collecte est presque saturé pour les scénarios avec trois agents, la baisse des performances globales d'atterrissage diminue le ratio de collecte et les performances d'atterrissage. En général, il est évident que l'approche proposée ne peut pas seulement généraliser sur toute la gamme des paramètres de scénario rencontrés pendant la formation, mais peut également extrapoler avec succès aux paramètres hors distribution.

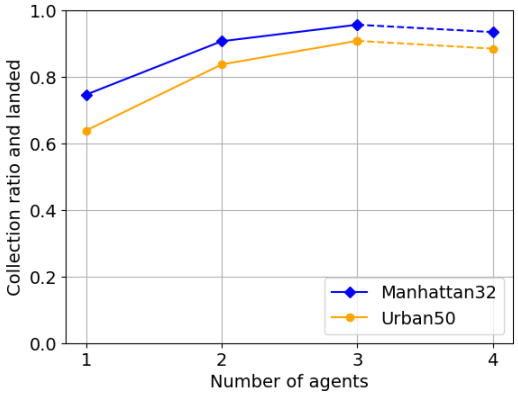
A.5 Planification du chemin de couverture (Chapitre 6)

En plus de la planification de trajectoire pour les applications dans les réseaux assistés par drones, les méthodes DRL présentées dans les chapitres précédents peuvent également être utilisées dans d'autres domaines où les drones autonomes présentent un intérêt. Un exemple pour ces applications est la planification de trajectoire de couverture de zone (CPP) [78], un problème de robotique classique. Comme son nom l'indique, la couverture de tous les points à l'intérieur d'une zone d'intérêt avec CPP est liée à la planification de chemin conventionnelle, où le but est de trouver un chemin entre les positions de départ et d'arrivée. En général, le CPP vise à couvrir la plus grande partie possible de la zone cible en respectant des contraintes d'énergie ou de longueur de chemin données, tout en évitant les obstacles ou les zones interdites de vol. Comme les méthodes DRL proposées sont très flexibles, elles peuvent fondamentalement être employées à n'importe quel problème de planification de trajectoire avec un objectif d'optimisation supplémentaire, par exemple la maximisation des données collectées à partir des dispositifs IoT ou la couverture complète d'une zone d'intérêt. Dans ce chapitre, nous démontrons l'applicabilité de notre approche de planification DDQN basée sur les cartes à ces deux scénarios de mission nettement différents. L'influence des principaux paramètres de traitement des cartes sur les performances d'apprentissage des trajectoires est également analysée.

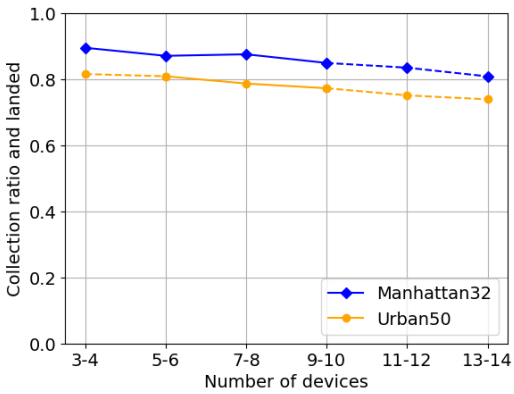
A.6 Planification de trajectoire des drones assistée par modèle (Chapitre 7)

Dans ce chapitre, nous étudions une autre approche qui s'attaque à la forte demande en données d'entraînement des méthodes DRL pour la planification de trajectoires de drones. Nous proposons ici une approche d'apprentissage Q profond assistée par un modèle qui, contrairement aux travaux précédents, réduit considérablement le besoin de données d'entraînement importants, tout en atteignant l'objectif primordial de l'apprentissage Q profond, à savoir guider un drone à batterie limitée vers une trajectoire efficace de collecte de données, sans connaissance préalable des caractéristiques des canaux sans fil ni connaissance limitée de l'emplacement des nœuds IoT sans fil. L'idée clé consiste à utiliser un petit sous-ensemble de nœuds comme ancres (c'est-à-dire avec une localisation connue) et à apprendre un modèle de l'environnement de propagation tout en estimant

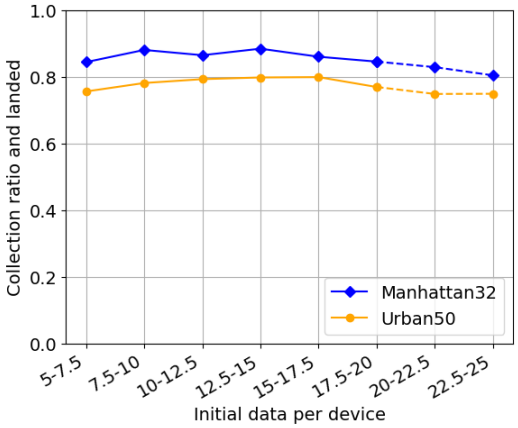
implicitement les positions des nœuds réguliers. Nous montrons que par rapport aux approches DRL standard, l'approche proposée, assistée par un modèle, nécessite au moins un ordre de grandeur de moins de données d'entraînement pour atteindre des performances de collecte de données identiques, offrant ainsi une première étape pour faire de la DRL une solution viable au problème.



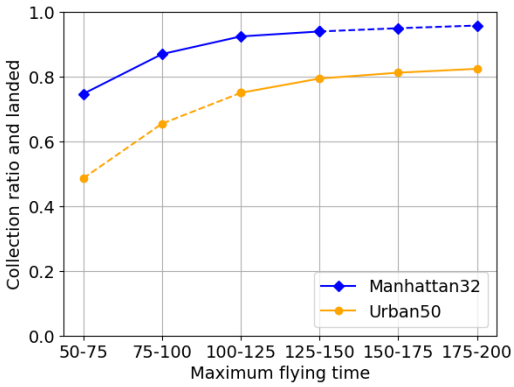
(a) Nombre d'agents $I \in \{1, 2, 3, 4\}$.



(b) Nombre de capteurs IoT $K \in [3, 14]$ triés en tranches de deux.



(c) Données à collecter auprès des capteurs $D_{k,init} \in [5, 25]$ triés en huit bacs.



(d) Temps de vol maximal trié en six catégories dans $b_0 \in [50, 200]$.

FIGURE A.5 – Influence des paramètres spécifiques du scénario sur le ratio de collecte de données avec atterrissage réussi de tous les agents. Chaque point de données est une moyenne de 500 itérations de Monte Carlo sur les espaces de paramètres respectifs pour les cartes 'Manhattan32' et 'Urban50'. Les paramètres dans la plage d'entraînement sont représentés par des lignes pleines et l'évaluation des paramètres hors distribution par des lignes pointillées.

Bibliography

[1] Y. Zeng, Q. Wu, and R. Zhang, "Accessing from the sky: A tutorial on UAV communications for 5G and beyond," *Proceedings of the IEEE*, vol. 107, no. 12, pp. 2327–2375, 2019.

[2] Y. Li, "Deep reinforcement learning," *arXiv:1810.06339 [cs.LG]*, 2018.

[3] R. Trumpp, "Deep reinforcement learning-based coordination of autonomous cars and street-crossing pedestrians," Master's thesis, Technical University of Munich, Germany, May 2021.

[4] Federal Aviation Authority (FAA), "New FAA rules for small unmanned aircraft systems go into effect - [press release]," August 2016. [Online]. Available: https://www.faa.gov/news/press_releases/news_story.cfm?newsId=20734

[5] L. Wood, "$63.6 bn drone service markets, 2025 - increasing use of drone services for industry-specific solutions - [online]," *Businesswire*, 17 Apr 2019. [Online]. Available: https://www.businesswire.com/news/home/20190417005302/en

[6] R. Hallion, *Taking flight: inventing the aerial age, from antiquity through the First World War*. Oxford University Press, 2003.

[7] P. E. Ross, "Iceland's consumers try drone delivery - [online]," *IEEE Spectrum*, vol. 55, no. 10, pp. 12–13, 2018.

[8] M. Minevich, "How Japan is tackling the national & global infrastructure crisis & pioneering social impact - [online]," *Forbes*, 21 Apr 2020. [Online]. Available: https://www.forbes.com/sites/markminevich/2020/04/21/how-japan-is-tackling-the-national--global-infrastructure-crisis--pioneering-social-impact/#483791ef2eaf

[9] K. Namuduri, "Flying cell towers to the rescue," *IEEE Spectrum*, vol. 54, no. 9, pp. 38–43, 2017.

[10] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[11] Y. Zeng, J. Lyu, and R. Zhang, "Cellular-connected UAV: Potential, challenges, and promising technologies," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 120–127, 2018.

[12] Amazon Inc., "Amazon Prime Air delivery service." [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011

[13] E. Ackerman, "Wing officially launches Australian drone delivery service," *IEEE Spectrum*, October 2019. [Online]. Available: https://spectrum.ieee.org/automaton/robotics/drones/wing-officially-launches-australian-drone-delivery-service

[14] B. Van der Bergh, A. Chiumento, and S. Pollin, "LTE in the sky: Trading off propagation benefits with interference costs for aerial nodes," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 44–50, 2016.

[15] Qualcomm, "LTE unmanned aircraft systems trial report." Tech. Rep., May 2017. [Online]. Available: https://www.qualcomm.com/documents/lte-unmanned-aircraft-systems-trial-report

[16] 3GPP TR 36.777 version 15.0.0 Release 15, "Technical specification group radio access network: Study on enhanced LTE support for aerial vehicles," 3GPP, Tech. Rep., January 2018.

[17] 3GPP TS 22.125 and TS 22.261 Release 17, "5G enhancement for UAVs," 3GPP, Tech. Rep., upcoming.

[18] U. Challita, W. Saad, and C. Bettstetter, "Interference management for cellular-connected UAVs: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2125–2140, 2019.

[19] O. Esrafilian, R. Gangula, and D. Gesbert, "3D-Map assisted UAV trajectory design under cellular connectivity constraints," in *IEEE International Conference on Communications (ICC)*. IEEE, 2020.

[20] R. Amer, W. Saad, and N. Marchetti, "Mobility in the sky: Performance and mobility analysis for cellular-connected UAVs," *IEEE Transactions on Communications*, vol. 68, no. 5, pp. 3229–3246, 2020.

[21] J. Stanczak, I. Z. Kovacs, D. Koziol, J. Wigard, R. Amorim, and H. Nguyen, "Mobility challenges for unmanned aerial vehicles connected to cellular LTE networks," in *IEEE 87th Vehicular Technology Conference (VTC Spring)*. IEEE, 2018.

[22] A. Azari, F. Ghavimi, M. Ozger, R. Jantti, and C. Cavdar, "Machine learning assisted handover and resource management for cellular connected drones," in *IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020.

[23] A. Fakhreddine, C. Bettstetter, S. Hayat, R. Muzaffar, and D. Emini, "Handover challenges for cellular-connected drones," in *Proceedings of the 5th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, 2019, pp. 9–14.

[24] J. Hu, H. Zhang, L. Song, Z. Han, and H. V. Poor, "Reinforcement learning for a cellular internet of UAVs: Protocol design, trajectory control, and resource management," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 116–123, 2020.

[25] B. Galkin, E. Fonseca, R. Amer, L. A. DaSilva, and I. Dusparic, "REQIBA: Regression and deep q-learning for intelligent UAV cellular user to base station association," *arXiv:2010.01126 [cs.IT]*, 2020.

[26] G. Geraci, A. Garcia-Rodriguez, M. M. Azari, A. Lozano, M. Mezzavilla, S. Chatzinotas, Y. Chen, S. Rangan, and M. D. Renzo, "What will the future of UAV cellular communications be? A flight from 5G to 6G," *2105.04842 [cs.IT]*, 2021.

[27] S. Rohde, M. Putzke, and C. Wietfeld, "Ad hoc self-healing of ofdma networks using UAV-based relays," *Ad Hoc Networks*, vol. 11, no. 7, pp. 1893–1906, 2013.

[28] V. Sharma, K. Srinivasan, H.-C. Chao, K.-L. Hua, and W.-H. Cheng, "Intelligent deployment of UAVs in 5G heterogeneous communication environment for improved coverage," *Journal of Network and Computer Applications*, vol. 85, pp. 94–105, 2017.

[29] H. Bayerlein, P. De Kerret, and D. Gesbert, "Trajectory optimization for autonomous flying base station via reinforcement learning," in *IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018.

[30] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Distributed multi-agent meta learning for trajectory design in wireless drone networks," *to appear in IEEE Journal on Selected Areas in Communications, arXiv:2012.03158 [cs.LG]*, 2020.

[31] W. Guo, C. Devine, and S. Wang, "Performance analysis of micro unmanned airborne communication relays for cellular networks," in *9th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP)*. IEEE, 2014, pp. 658–663.

[32] D. Orfanus, E. P. de Freitas, and F. Eliassen, "Self-organization as a supporting paradigm for military UAV relay networks," *IEEE Communications Letters*, vol. 20, no. 4, pp. 804–807, 2016.

[33] J. Tang, J. Song, J. Ou, J. Luo, X. Zhang, and K.-K. Wong, "Minimum throughput maximization for multi-UAV enabled WPCN: A deep reinforcement learning method," *IEEE Access*, vol. 8, pp. 9124–9132, 2020.

[34] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile unmanned aerial vehicles (uavs) for energy-efficient internet of things communications," *IEEE Transactions on Wireless Communications*, vol. 16, no. 11, pp. 7574–7589, 2017.

[35] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "Multi-UAV path planning for wireless data harvesting with deep reinforcement learning," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1171–1187, 2021.

[36] W. Saad, M. Bennis, M. Mozaffari, and X. Lin, *Wireless Communications and Networking for Unmanned Aerial Vehicles*. Cambridge University Press, 2020.

[37] W. Khawaja, I. Guvenc, D. W. Matolak, U.-C. Fiebig, and N. Schneckenberger, "A survey of air-to-ground propagation channel modeling for unmanned aerial vehicles," in *UAV Communications for 5G and Beyond*, Y. Zeng, I. Guvenc, R. Zhang, G. Geraci, and D. W. Matolak, Eds. Wiley Online Library, 2020, ch. 2.

[38] A. Merwaday and I. Guvenc, "UAV assisted heterogeneous networks for public safety communications," in *IEEE wireless communications and networking conference workshops (WCNCW)*. IEEE, 2015, pp. 329–334.

[39] Q. Zhang, W. Saad, M. Bennis, X. Lu, M. Debbah, and W. Zuo, "Predictive deployment of UAV base stations in wireless networks: Machine learning meets contract theory," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 637–652, 2021.

[40] Y. Zeng and R. Zhang, "Energy-efficient uav communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[41] R. Gangula, P. de Kerret, O. Esrafilian, and D. Gesbert, "Trajectory optimization for mobile access point," in *51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2017, pp. 1412–1416.

[42] C. H. Liu, X. Ma, X. Gao, and J. Tang, "Distributed energy-efficient multi-UAV navigation for long-term communication coverage by deep reinforcement learning," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1274–1285, 2020.

[43] E. Yanmaz, R. Kuschnig, and C. Bettstetter, "Channel measurements over 802.11 a-based UAV-to-ground links," in *IEEE GLOBECOM Workshops*. IEEE, 2011, pp. 1280–1284.

[44] A. I. Hentati and L. C. Fourati, "Comprehensive survey of UAVs communication networks," *Computer Standards & Interfaces*, p. 103451, 2020.

[45] Y. Zeng, I. Guvenc, R. Zhang, G. Geraci, and D. W. Matolak, *UAV Communications for 5G and Beyond*. Wiley Online Library, 2020.

[46] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. G. Giordano, A. Garcia-Rodriguez, and J. Yuan, "Survey on UAV cellular communications: Practical aspects, standardization advancements, regulation, and security challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3417–3442, 2019.

[47] M. Harris, "Facebook cancels program to deliver internet by aquila drones," *IEEE Spectrum*, June 2018. [Online]. Available: https://spectrum.ieee.org/tech-talk/telecom/internet/facebook-pulls-out-of-secret-spaceport-internet-drone-tests

[48] M. Koziol, "Alphabet's loon failed to bring internet to the world. what went wrong?" *IEEE Spectrum*, February 2021. [Online]. Available: https://spectrum.ieee.org/tech-talk/telecom/wireless/why-did-alphabets-loon-fail-to-bring-internet-to-the-world

[49] E. U. A. S. A. (EASA), "Open category - civil drones," Tech. Rep., December 2021. [Online]. Available: https://www.easa.europa.eu/domains/civil-drones-rpas/open-category-civil-drones

[50] F. A. Administration, "Executive summary: Final rule on operation of small unmanned aircraft systems over people," Tech. Rep., December 2020. [Online]. Available: https://www.faa.gov/news/media/attachments/OOP_Executive_Summary.pdf

[51] K. Dalamagkidis, "Classification of UAVs," in *Handbook of unmanned aerial vehicles*, K. P. Valavanis and G. J. Vachtsevanos, Eds. Springer, 2015, ch. 10.

[52] Z. Ullah, F. Al-Turjman, and L. Mostarda, "Cognition in UAV-aided 5G and beyond communications: A survey," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 3, pp. 872–891, 2020.

[53] Institut national de l'information géographique et forestière, "Restrictions UAS catégorie ouverte et aéromodélisme," *Géoportail*, May 2021. [Online]. Available: https://www.geoportail.gouv.fr/carte?c=0.41206420680184447,46.197314358870585&z=6&l0=ORTHOIMAGERY.ORTHOPHOTOS::GEOPORTAIL:OGC:WMTS(1)&l1=TRANSPORTS.DRONES.RESTRICTIONS::GEOPORTAIL:OGC:WMTS(1)&permalink=yes

[54] Federal Aviation Authority (FAA), "Unmanned aircraft systems beyond program," October 2020. [Online]. Available: https://www.faa.gov/uas/programs_partnerships/beyond

[55] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[56] R. Bellman, *An introduction to artificial intelligence: can computers think?* Boyd & Frase, 1978.

[57] T. M. Mitchell *et al.*, "Machine learning," 1997.

[58] K. Xiao, J. Zhao, Y. He, and S. Yu, "Trajectory prediction of UAV in smart city using recurrent neural networks," in *IEEE International Conference on Communications (ICC)*. IEEE, 2019.

[59] Y. Zhang, J. Wen, G. Yang, Z. He, and X. Luo, "Air-to-air path loss prediction based on machine learning methods in urban environments," *Wireless Communications and Mobile Computing*, vol. 2018.

[60] R. Amorim, J. Wigard, H. Nguyen, I. Z. Kovacs, and P. Mogensen, "Machine-learning identification of airborne UAV-UEs based on LTE radio measurements," in *IEEE GLOBECOM Workshops*. IEEE, 2017.

[61] M.-A. Lahmeri, M. A. Kishk, and M.-S. Alouini, "Artificial intelligence for UAV-enabled wireless networks: A survey," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1015–1040, 2021.

[62] H. Peng, A. Razi, F. Afghah, and J. Ashdown, "A unified framework for joint mobility prediction and object profiling of drones in UAV networks," *Journal of Communications and Networks*, vol. 20, no. 5, pp. 434–442, 2018.

[63] J.-L. Wang, Y.-R. Li, A. B. Adege, L.-C. Wang, S.-S. Jeng, and J.-Y. Chen, "Machine learning based rapid 3d channel modeling for UAV communication networks," in *16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2019.

[64] T. M. Hoang, N. M. Nguyen, and T. Q. Duong, "Detection of eavesdropping attack in UAV-aided wireless systems: Unsupervised learning with one-class svm and k-means clustering," *IEEE Wireless Communications Letters*, vol. 9, no. 2, pp. 139–142, 2019.

[65] S. Khan, C. F. Liew, T. Yairi, and R. McWilliam, "Unsupervised anomaly detection in unmanned aerial vehicles," *Applied Soft Computing*, vol. 83, p. 105650, 2019.

[66] C. H. Liu, Z. Dai, Y. Zhao, J. Crowcroft, D. O. Wu, and K. Leung, "Distributed and energy-efficient mobile crowdsensing with charging stations by deep reinforcement learning," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 130–146, 2021.

[67] J. Cui, Z. Ding, Y. Deng, and A. Nallanathan, "Model-free based automated trajectory optimization for UAVs toward data transmission," in *IEEE Global Communications Conference (GLOBECOM)*, 2019.

[68] Q. Li, M. Ding, C. Ma, C. Liu, Z. Lin, and Y.-C. Liang, "A reinforcement learning based user association algorithm for UAV networks," in *28th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2018.

[69] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv:1912.04977 [cs.LG]*, 2019.

[70] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for UAVs-enabled wireless networks: Use cases, challenges, and open problems," *IEEE Access*, vol. 8, pp. 53 841–53 849, 2020.

[71] P. S. Bithas, E. T. Michailidis, N. Nomikos, D. Vouyioukas, and A. G. Kanatas, "A survey on machine-learning techniques for UAV-based communications," *Sensors*, vol. 19, no. 23, p. 5170, 2019.

[72] X. Li, H. Yao, J. Wang, S. Wu, C. Jiang, and Y. Qian, "Rechargeable multi-UAV aided seamless coverage for QoS-guaranteed IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 902–10 914, 2019.

[73] X. Liu, Y. Liu, and Y. Chen, "Reinforcement learning in multiple-UAV networks: Deployment and movement design," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8036–8049, 2019.

[74] R. Shakeri, M. A. Al-Garadi, A. Badawy, A. Mohamed, T. Khattab, A. K. Al-Ali, K. A. Harras, and M. Guizani, "Design challenges of multi-UAV systems in cyber-physical applications: A comprehensive survey and future directions," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3340–3385, 2019.

[75] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *International Conference on Machine Learning Workshop on Real-Life RL, arXiv:1904.12901 [cs.LG]*, 2019.

[76] H. Bayerlein, R. Gangula, and D. Gesbert, "Learning to rest: A Q-learning approach to flying base station trajectory design with landing spots," in *52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2018, pp. 724–728.

[77] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "UAV path planning for wireless data harvesting: A deep reinforcement learning approach," in *IEEE Global Communications Conference (GLOBECOM)*, 2020.

[78] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "UAV coverage path planning under varying power constraints using deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[79] ——, "UAV path planning using global and local map information with deep reinforcement learning," *submitted to 20th International Conference on Advanced Robotics (ICAR), arXiv:2010.06917 [cs.RO]*, 2021.

[80] O. Esrafilian, H. Bayerlein, and D. Gesbert, "Model-aided deep reinforcement learning for sample-efficient UAV trajectory design in IoT networks," *to be presented at IEEE Global Communications Conference (GLOBECOM), arXiv:2104.10403 [cs.IT]*, 2021.

[81] Z. Liu, R. Sengupta, and A. Kurzhanskiy, "A power consumption model for multi-rotor small unmanned aircraft systems," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017.

[82] N. Ahmed, S. S. Kanhere, and S. Jha, "On the importance of link characterization for aerial wireless sensor networks," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 52–57, 2016.

[83] R. S. Sutton and A. G. Barto, *Reinforcement Learning: an introduction*, 2nd ed. MIT Press, 2018.

[84] M. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, 2012.

[85] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2017.

[86] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2005.

[87] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge University, 1989.

[88] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[89] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[90] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 4, no. 1, pp. 1–103, 2010.

[91] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[92] J. Kosmerl and A. Vilhar, "Base stations placement optimization in wireless networks for emergency communications," in *IEEE International Conference on Communications Workshops (ICC)*, 2014, pp. 200–205.

[93] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Optimal transport theory for power-efficient deployment of unmanned aerial vehicles," in *IEEE International Conference on Communications (ICC)*, 2016.

[94] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, 2018.

[95] R. Gangula, D. Gesbert, D.-F. Külzer, and J. M. Franceschi Quintero, "A landing spot approach to enhancing the performance of UAV-aided wireless networks," in *IEEE Workshop on Integrating UAVs into 5G, International Conference on Communications (ICC)*, 2018.

[96] (2017) Eurecom - autonomous aerial cellular relaying robots (video). [Online]. Available: https://youtu.be/GI_lOsg_qmQ

[97] C. Q. Choi, "New electric drone has groundbreaking flight time - [online]," *IEEE Spectrum*, Sep 2018.

[98] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Communications Surveys & Tutorials*, 2021.

[99] B. Zhang, C. H. Liu, J. Tang, Z. Xu, J. Ma, and W. Wang, "Learning-based energy-efficient data collection by unmanned vehicles in smart cities," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1666–1676, 2017.

[100] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980 [cs.LG]*, 2014.

[101] D. P. Bertsekas, *Dynamic programming and optimal control*, 3rd ed. Athena Scientific Belmont, MA, 2011, vol. 2.

[102] M. Yi, X. Wang, J. Liu, Y. Zhang, and B. Bai, "Deep reinforcement learning for fresh data collection in UAV-assisted IoT networks," in *IEEE Conference on Computer Communications Workshops (IEEE INFOCOM WKSHPS)*, 2020.

[103] H. Qi, Z. Hu, H. Huang, X. Wen, and Z. Lu, "Energy efficient 3-D UAV control for persistent communication service and fairness: A deep reinforcement learning approach," *IEEE Access*, vol. 8, 2020.

[104] O. Esrafilian, R. Gangula, and D. Gesbert, "Learning to communicate in UAV-aided wireless networks: Map-based approaches," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1791–1802, 2018.

[105] C. You and R. Zhang, "Hybrid offline-online design for UAV-enabled data harvesting in probabilistic LoS channel," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 3753–3768, 2020.

[106] Z. Zhou, J. Feng, B. Gu, B. Ai, S. Mumtaz, J. Rodriguez, and M. Guizani, "When mobile crowd sensing meets UAV: Energy-efficient task assignment and route planning," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5526–5538, 2018.

[107] B. G. Maciel-Pearson, L. Marchegiani, S. Akcay, A. Atapour-Abarghouei, J. Garforth, and T. P. Breckon, "Online deep reinforcement learning for autonomous UAV navigation and exploration of outdoor environments," *arXiv:1912.05684 [cs.CV]*, 2019.

[108] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016, pp. 2094–2100.

[109] S. Zhang and R. S. Sutton, "A deeper look at experience replay," *arXiv:1712.01275 [cs.LG]*, 2017.

[110] 3GPP TR 38.901 version 14.0.0 Release 14, "Study on channel model for frequencies from 0.5 to 100 GHz," ETSI, Tech. Rep., May 2017.

[111] Y. Pan, Y. Yang, and W. Li, "A deep learning trained by genetic algorithm to improve the efficiency of path planning for data collection with multi-UAV," *IEEE Access*, vol. 9, pp. 7994–8005, 2021.

[112] Y. Zhang, Z. Mou, F. Gao, L. Xing, J. Jiang, and Z. Han, "Hierarchical deep reinforcement learning for backscattering data collection with multiple UAVs," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3786–3800, 2021.

[113] F. Wu, H. Zhang, J. Wu, L. Song, Z. Han, and H. V. Poor, "UAV-to-device underlay communications: Age of information minimization by multi-agent deep reinforcement learning," *IEEE Transactions on Communications (early access)*, 2021.

[114] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, "Cooperative Internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 6807–6821, 2020.

[115] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad, "Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks," in *IEEE Global Communications Conference (GLOBECOM)*, 2019.

[116] F. Venturini, F. Mason, F. Pase, F. Chiariotti, A. Testolin, A. Zanella, and M. Zorzi, "Distributed reinforcement learning for flexible UAV swarm control with transfer learning capabilities," in *Proceedings of the 6th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, 2020.

[117] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[118] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.

[119] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[120] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *arXiv:1911.10635 [cs.LG], to be published as chapter in Handbook on RL and Control (Springer)*, 2019.

[121] M. Zhang, C. Dong, and Y. Huang, "FS-MAC: An adaptive MAC protocol with fault-tolerant synchronous switching for FANETs," *IEEE Access*, vol. 7, pp. 80 602–80 613, 2019.

[122] Link Technologies, "ANT-5GWWS1-SMA cellular sub-6 antenna datasheet," Tech. Rep. [Online]. Available: https://eu.mouser.com/datasheet/2/238/ant-5gwws1-sma-ds-1855874.pdf

[123] B. Partov, D. J. Leith, and R. Razavi, "Utility fair optimization of antenna tilt angles in LTE networks," *IEEE/ACM Transactions on Networking*, vol. 23, no. 1, pp. 175–185, 2015.

[124] Kathrein, "Multi-band F-Panel dual polarization 742215 antenna datasheet," Tech. Rep. [Online]. Available: http://www.selteq.com/Products/kathrein/data_sheets/742215.pdf

[125] M. Samir, M. Elhattab, C. Assi, S. Sharafeddine, and A. Ghrayeb, "Optimizing age of information through aerial reconfigurable intelligent surfaces: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 4, pp. 3978–3983, 2021.

[126] R. Zhong, X. Liu, Y. Liu, and Y. Chen, "Multi-agent reinforcement learning in noma-aided uav networks for cellular offloading," *IEEE Transactions on Wireless Communications*, 2021.

[127] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[128] T. Cabreira, L. Brisolara, and P. R Ferreira, "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, 2019.

[129] D. Gimesy, "Drones and thermal imaging: saving koalas injured in the bushfires - [online]," *The Guardian*, 10 Feb 2020. [Online]. Available: https://www.theguardian.com/australia-news/gallery/2020/feb/11/drones-thermal-imaging-australia-koalas-bushfire-crisis

[130] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1-2, pp. 25–50, 2000.

[131] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and service robotics*. Springer, 1998, pp. 203–209.

[132] R. Mannadiar and I. Rekleitis, "Optimal coverage of a known arbitrary environment," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 5525–5530.

[133] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Optimal complete terrain coverage using an unmanned aerial vehicle," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 2513–2519.

[134] J. Xie, L. R. G. Carrillo, and L. Jin, "An integrated traveling salesman and coverage path planning problem for unmanned aircraft systems," *IEEE control systems letters*, vol. 3, no. 1, pp. 67–72, 2018.

[135] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 718–724, 2004.

[136] C. Piciarelli and G. L. Foresti, "Drone patrolling with reinforcement learning," in *Proceedings of the 13th International Conference on Distributed Smart Cameras*. ACM, 2019. [Online]. Available: https://doi.org/10.1145/3349801.3349805

[137] K. D. Julian and M. J. Kochenderfer, "Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 8, pp. 1768–1778, 2019.

[138] E. Seraj and M. Gombolay, "Coordinated control of UAVs for human-centered active sensing of wildfires," in *American Control Conference (ACC)*. IEEE, 2020, pp. 1845–1852.

[139] D. Baldazo, J. Parras, and S. Zazo, "Decentralized multi-agent deep reinforcement learning in swarms of drones for flood monitoring," in *27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019.

[140] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[141] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *International Conference on Machine Learning (ICML)*, 2016.

[142] N. Heess, G. Wayne, D. Silver, T. Lillicrap, Y. Tassa, and T. Erez, "Learning continuous control policies by stochastic value gradients," in *28th International Conference on Neural Information Processing Systems*, 2015.

[143] X. Li, Q. Wang, J. Liu, and W. Zhang, "Trajectory design and generalization for UAV enabled networks: A deep reinforcement learning approach," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2020.

[144] O. Esrafilian, R. Gangula, and D. Gesbert, "Three-dimensional-map-based trajectory design in UAV-aided wireless localization systems," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9894–9904, 2021.

[145] J. Chen, U. Yatnalli, and D. Gesbert, "Learning radio maps for UAV-aided wireless networks: A segmented regression approach," in *IEEE International Conference on Communications (ICC)*, 2017.

[146] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks (ICNN)*, 1995.

[147] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[148] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering (early access)*, 2021.

[149] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv:2005.01643 [cs.LG]*, 2020.

[150] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.