esprit
Se former autrement

2019 - 2020
GRADUATION PROJECT

NATIONAL ENGINEERING DEGREE

SPECIALTY : INFORMATION TECHNOLOGY

Offline and Online UAV-enabled Data Collection in Time-constrained IoT Networks

By: *Oussama Ghdiri*

Academic supervisor: *Dr. Jihene Ben Abderrazak*

Corporate Internship Supervisor: *Dr. Wael Jaafar*

Carleton UNIVERSITY

cdio

EUR-ACE

Cti

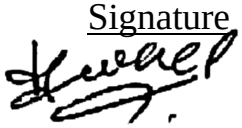
CONFÉRENCE DES GRANDES ÉCOLES

Carleton UNIVERSITY

I hereby authorize the student to submit his internship report in the aim of a defense.

Dr. Wael Jaafar, Corporate Internship Supervisor

Signature



Wael Jaafar, PhD
NSERC Research Fellow,
Department of Systems and Computer Engineering,
Carleton University,

esprit
Se former autrement

I hereby authorize the student to submit his internship report in the aim of a defense.

Dr. Jihene Ben Abderrazak, Academic Supervisor

Signature

ABSTRACT

Recently, unmanned aerial vehicle (UAV) technology is endorsed to enable applications in domains like Internet of things, wireless sensor networks, and cellular networks. Particularly, time-sensitive and energy-limited IoT networks located in hard-to-reach areas require efficient/cost-effective data collection solutions.

To address this matter, we consider a multi-UAV enabled IoT network, where several UAVs collect data from time-constrained sensor nodes (SNs), i.e., within data deadlines. In our framework, SNs are managed by cluster heads (CHs), then multiple UAVs collect data from the latter. We formulate the maximization problem of the amount of data collected within the deadlines, while reducing the UAVs' energy consumption, subject to communication, UAVs mission time and battery capacity constraints.

To solve it, we propose a two-step approach. In the first step, an efficient K-means based method groups SNs and deploys CHs. Then, UAV-based offline and online data collection methods are proposed. In the offline setting where the system's status is known beforehand, UAV paths are determined using near-optimal metaheuristics. Through simulations, we demonstrate that Tabu search achieves the best performances. In the online setting where no system information is available, deep reinforcement learning (DRL) based approaches are proposed. Results prove the superiority of the actor-critic DRL algorithm compared to other benchmarks.

Keywords : Unmanned Aerial Vehicle, UAV, Internet of Things, IoT, Clustering, Vehicle Routing Problem, Reinforcement Learning, Data collection.

DEDICATIONS

I dedicate this work

***To my parents**, who have been a constant source of inspiration and whose good examples have taught me to work hard for the things that I aspire to achieve, your prays were enlightening my way. The endless support, unconditional love and privileges that you have given me throughout my entire life can not be stressed enough.*

***To my family members, friends and colleagues,** who have been a true motivation and driving force for all my achievements and actions.*

***To Dr. Jihene Ben Abderrazak and Dr. Wael Jaafar**, who gave me the opportunity of working on an interesting project and teaching me a great deal about how to do research. Not every student is lucky enough to work under the supervision of people with such knowledge and passion like you. I cannot thank you enough for inspiring me, helping to me and bringing out the best in me. I hope that what comes next for you is as great as you are.*

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep and sincere gratitude to my supervisor **Dr. Jihene Ben Abderrazak** for her invaluable help, precious advice and endless support. Without her time, efforts, guidance and inspiring encouragement this project would not have been possible. I could not have had a better advisor for my graduation project.

I could not go any further without acknowledging the tremendous contributions of **Dr. Wael Jaafar** for the time and efforts dedicated to assessing this project. His reviews and comments were incredibly helpful and very much appreciated.

Further, I am very grateful to **Dr. Safwan Alfattani** and **Prof. Halim Yanikomeroglu** for the interesting and inspiring discussions and meetings.

Besides the people already mentioned, I would like to thank those people that in one way or another have contributed to this work.

Last but not the least, my most respectful gratitude is addressed to the members of Jury who agreed to evaluate my work.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AC	Actor Critic
CH	Cluster Head
CVRPTW	Capacitated Vehicle Routing Problem With Time Windows
DQN	Deep Q-Network
DDQN	Double Deep Q-Network
DRL	Deep Reinforcement Learning
GLS	Guided Local Search
HA	Hovering Above
HAC	Hierarchical Agglomerative Clustering
HR	Hovering within a Range
IoT	Internet of Things
LoS	Line of Sight
MDP	Markov Decision Process
ML	Machine Learning
NLoS	Non-Line of Sight
RL	Reinforcement Learning
SA	Simulated Annealing
SARSA	State-Action-Reward-State-Action
SN	Sensor Node
TD	Temporal Difference
UAV	Unmanned Aerial Vehicle
VRP	Vehicle Routing Problem
WSN	Wireless Sensors Network

LIST OF SYMBOLS

(a, b)	Environment parameters
A	UAV rotor disc area
$\mathcal{A}_c$	Set of SNs associated with CH c
d_0	UAV fuselage drag ratio
d_{cu}	Distance between CH c and UAV u
d_{sc}	Distance between SN s and CH c
d_{safe}	Minimum safety distance between flying UAVs
d_{th}	Distance-based threshold for SN-CH link
$E_u(v, T)$	Consumed energy by UAV u during period T
F	Maximum number of SNs in a cluster
f_c	Carrier frequency
$\mathcal{K}$, K	Set and number of CHs
$\mathcal{K}_u$, K_u	Set and number of ordered CHs to visit by UAV u
$\mathcal{L}$	Set of CHs locations
$\mathcal{L}_u$	Set of ordered hovering locations for UAV u
l_{cu}^{LoS}	LoS path-loss for link between CH c and UAV u
l_{cu}^{NLoS}	NLoS path-loss for link between CH c and UAV u
$\mathcal{M}$, M	Set and number of SNs
P_{CH}	Transmit power of any CH
P_{cu}	Received power at UAV u from CH c
P_{SN}	Transmit power of any SN
$P_{\text{hov,u}}$	Hovering power of UAV u
$P_{\text{prop,u}}$	Propulsion power of UAV u

Pr_{cu}^{LoS}	LoS probability of link between CH c and UAV u
Pr_{cu}^{NLoS}	NLoS probability of link between CH c and UAV u
Q_c	Number of data packets to be transmitted by CH c
$\mathbf{q}^0$	3D location of the UAVs dockstation
$\mathbf{q}_i$	3D location of device i
R_{cu}	Data rate of the link between CH c and UAV u
r	UAV rotor solidity
S_p	Size of any packet
$S_u(t)$	Available battery capacity for UAV u at time t
$S_{\min}$	Safety battery capacity for any UAV
T_{cu}	Data collection time by UAV u from CH c
$T_{d,c}$	Time deadline of data stored in CH c
$T_{f,u}(c, c')$	Flight time of UAV u between CHs c and c'
$\mathcal{U}$, U	Set and number of available UAVs
U_{tip}	The tip speed of UAV's rotor blade
V	Speed of light
v_0	UAV mean rotor induced velocity
v_u	UAV u displacement speed
$v_{\max}$	Maximal UAV displacement speed
W	Bandwidth of the communication channel
α	Path-loss exponent
β_{LoS}	LoS excessive path-loss coefficient
β_{NLoS}	NLoS excessive path-loss coefficient
δ	Length of a time slot
γ_{sc}	Received signal-to-noise ratio at CH c
γ_{th}	Signal-to-noise ratio threshold for a SN-CH link
Λ_{cu}	Average path-loss of link between CH c and UAV u
σ^2	Noise power
ψ	Air density
θ_{cu}	Elevation angle between UAV u and CH c
ζ_B	UAV blade profile power
ζ_I	UAV induced power

LIST OF ALGORITHMS

GENERAL INTRODUCTION

In the midst of fighting the same enemy, the whole world has faced new challenges impacting countries, communities, and individuals in countless ways. Dealing with the unforeseen obstacles, technology is playing a pivotal role in helping humanity react, recover, and thrive in the COVID-19 hit world. Moreover, the obligation to reduce human contact as a health precaution has boosted the use of unmanned aerial vehicles (UAVs), commonly known as drones, in different situations. For instance, UAVs have been deployed to deliver medical supplies, collect or dispatch lab samples, deliver daily necessities, or monitor social distancing. Also, UAVs have been used to combat the pandemic through emergency announcements, mass screening, and public places disinfecting [1]. The noticed UAV-based positive experiences is accelerating their deployment beyond the immediate use in times of crisis. Yet, the enthusiasm for deploying UAVs applications has skyrocketed, but their potential is not fully unleashed.

The pandemic has not only been the ultimate catalyst speeding up the trend of UAVs, it is also accelerating the adoption of Internet of Things (IoT) systems. Indeed, IoT has been gaining momentum in both the industry and research. It is projected to infiltrate almost every aspect of our lives, connecting millions of devices and people [2].

Typically, IoT sensor nodes (SNs) are made of constrained computing devices with embedded sensing and wireless communication capabilities to communicate with other nodes or transmit collected data to a remote location for storage or further processing [3]. They are low-cost and energy-limited sensing devices with short transmission ranges. These constraints raise several challenges, especially for critical massive machine type communication systems.

In fact, a major concern is on how to gather data from massive IoT systems without the latter becoming outdated. Moreover, given that IoT devices might be deployed in hard-to-reach areas and/or over moving targets, e.g., monitored groups or herds of animals, collecting data from these IoT nodes in a timely manner becomes very challenging.

In this context, the use of UAVs as flying data collectors has gained interest [4]. UAVs can gather data from dispersed and large-scale IoT sensors and forward it to a control center. Due to the UAVs' high motion flexibility and maneuverability, strong line-of-sight (LoS) communication channels with ground devices, and moderately low flying altitudes to reach low-power IoT devices, UAV-based data collection is seen as the trigger of future and new applications that set stringent quality-of-service (QoS) requirements [5]–[7]. Despite their great potential, the use of UAVs for data collection raises several challenges. Indeed, current UAV technology struggles with the limited on-board energy [8], thus often requiring the deployment of multiple UAVs to cover large-scale IoT fields and increasing the data collection costs. Moreover, depending on the criticality level of the IoT application linked to the data freshness and sensitivity, both the IoT network characteristics (e.g., IoT nodes' mobility, SNs' clustering, and deployment of cluster heads (CHs)), and the UAV characteristics (e.g., flying speed, communication capability), have an important impact on realizing the targeted QoS. Motivated by the aforementioned points, we propose here a practical data collection approach for large-scale IoT systems, where multiple UAVs collect sensed data with different time deadlines.

This work is structured as follows: In the first chapter, the general context of the project is presented. In chapter two, the theoretical background and some major concepts are described. Then, the following chapters presents the full process of our proposed approaches, introduces the basic mathematical formalism, the experiments, the most relevant and interesting results. In the conclusion, we present a summary of the core findings of this project and we provide some immediate extensions of our work. Finally, in the appendix, we present a copy of the accepted paper in IEEE GLOBECOM.

CHAPTER 1

GENERAL CONTEXT

Introduction

This first chapter starts with the general context of the project and a presentation of the internship setting. Then, the background, research motivation, objectives and related works are outlined. Finally, our contributions are identified.

1.1 Context of the internship

The internship project is the end of studies project required for obtaining the engineering degree at ESPRIT school of Engineering. The project is a collaboration between ESPRIT and Carleton University in Canada. Collaboration and links to external partners is an important cornerstone of research and innovation for any organisational actor. With the aim to further improve their external R&D links, Esprit-Tech has approached the department of Systems and Computer Engineering at Carleton University to conduct a research on innovation of artificial intelligence for wireless communications and networks.

1.1.1 Esprit-Tech

Research has been proved to be a crucial factor moving the world technological frontiers, while at the same time facilitating new technological and scientific innovations. ESPRIT has invested in research and has given it high priority since its creation. Therefore, a Research-Development-Innovation (RDI) department

called Esprit-Tech was created in 2010. Esprit-Tech attaches great importance to applied and innovative research that can have socio-economic benefits for enhancing productivity and efficiency in both the private and public sectors. Esprit-Tech is experiencing an increasing demand for R&D collaboration due to the increasing number of publications contributing to new knowledge across several disciplines over the last few years.

1.1.2 Carleton University

Carleton University is a comprehensive and research-intensive public university in Ottawa, Canada. Carleton has long been known as a university that promotes research excellence and connectedness, and enjoys partnerships around the globe. The university is known internationally as a recognized research leader that focuses both on tangible outcomes and the development of knowledge with longer-term impacts. Carleton works on a variety of research areas including computing systems, communication systems, multimedia, software engineering, biomedical engineering, and management of engineering processes. This combination of expertise makes it uniquely suited for exploring the new frontiers of technology integration and new ideas that have significant impact on these fields of study.

1.2 Background & motivation

The outbreak of COVID-19 crisis has shone a spotlight on areas of weakness, yet it has triggered creativity and innovation. Indeed, IoT technologies are at the heart of these innovations, enabling some tasks to be automated and work to be carried out remotely. Briefly, IoT has been a game changer not only in reducing casualties and keeping cities functional but also to ignite innovations in the near future.

Hence, recently, research efforts are channeled towards improving data collection in IoT applications. In addition, there has been a growing research interest in applying Unmanned Aerial Vehicle as flying data collectors in IoT networks. This work investigates the data collection process using UAVs, and proposes an efficient framework to gather data form time-constrained IoT networks in different scenarios using real-time and simulation-based scenarios. Accordingly, collecting data from massive and/or wide IoT fields using UAVs needs a careful design. Indeed, cost-effective data collection can be achieved through: 1) IoT network deployment optimization, and 2) efficient UAV deployment and path optimization.

For the IoT network deployment, it can be optimized through an efficient SNs clustering and strategic CHs placement. For UAVs deployment, a careful design of trajectories is required, either for the offline scenario where the system's status is available beforehand, or the online scenario where no or limited information is available for the UAVs' mission. The offline scenario corresponds to applications for dedicated or private IoT networks, where the initial IoT system design is fixed and changes occasionally. In this case, the trajectories can be pre-planned prior to deployment (i.e., offline), and they are only updated when sensors are added or removed from the initial system design. However, in situations where IoT sensors are mobile or their location information is fully or partially unavailable, offline path planning becomes unpractical. This corresponds to applications such as wildlife monitoring and intelligent transportation systems where IoT sensors may be mounted on animals and vehicles, respectively. Subsequently, it is crucial to adopt smart and energy-aware online path planning approaches to cope with dynamic environments.

To tackle the aforementioned issues, we propose a novel approaches for UAV-based data collection in time-constrained IoT networks. Specifically, we propose at first a K-means based algorithm to cluster SNs and deploy the minimum number of CHs. Then, using the smallest number of UAVs, energy-efficient offline and online multi-UAV path planning algorithms are presented. The objective is to collect the maximum amount of data with the smallest energy under requirements of data deadlines, communication conditions, UAVs' battery capacity, and UAVs' collision avoidance. To the best of our knowledge, this is the first work that investigates a complete UAV-based data collection framework for time-constrained IoT systems, where IoT network design and multi-UAV data collection strategies are optimized for offline and online scenarios. The UAVs path planning problem is formulated as a capacitated vehicle routing problem with time windows (CVRPTW) [9] and proven to be NP-hard. Hence, it is solved in the offline scenario using near-optimal meta-heuristic algorithms, namely Tabu search, simulated annealing (SA), and guided local search (GLS). However, for the online scenario, we propose reinforcement learning based algorithms, including deep-Q-network (DQN), double-DQN (DDQN), and actor-critic (AC), for efficient data collection.

1.3 State of the art

In this section, we present the related works that deal with IoT sensors clustering, and offline and online UAV path planning for data collection.

1.3.1 IoT sensors clustering

In large-scale IoT systems, collecting data from each of the SNs has proven to be energy inefficient due to the huge traffic load, increased energy consumption, and extended transmission delays. To circumvent this issue, IoT clustering is advocated as the most energy-efficient system design approach. Indeed, by putting SNs into small groups and deploying/designating CHs to manage their data, the traffic load towards the gateway (or collector) is drastically reduced and refined. Typically, CHs are IoT devices with higher communication ranges and processing capability than standard sensors. They are tasked with IoT data collection in a specific area, where they aggregate the received data, remove redundancy if any, and communicate the outcome to the gateway. The use of CHs becomes even more important when relying on UAVs for data collection.

Several works overviewed existing clustering techniques and discussed their potential [10]–[12]. Recently, adaptive clustering was proposed in [13], which takes into account the data distribution of SNs. In [14], Shao *et al.* proposed a cooperative framework with dynamic IoT clustering and power allocation using non-orthogonal multiple access. They claimed that their approach achieves fairness among SNs, reduces system complexity and delay, and increases the spectral efficiency. Also, a deep learning based clustering method was presented in [15], where the objective is to balance resource allocation among SNs and increase the network's life. Then, Hassan *et al.* proposed in [16] an energy-efficient clustering protocol based on the rotation of CHs' functions. Finally, in [4], Alfattani *et al.* proposed a simple, yet efficient K-means based clustering that deploys minimum number of CHs.

1.3.2 Offline UAV path planning

When IoT nodes have no direct links to fixed/permanent data collection gateways, UAVs can be deployed to collect data from SNs/CHs. Assuming that the locations of SNs/CHs are fixed and known a priori, the UAVs' trajectories can be preplanned.

In [4], the authors proposed a framework applicable in different environments for data collection from a clustered wireless sensor network using one or several UAVs. they aimed to minimize the mission costs in terms of number of clusters and traveled distance by UAVs, while guaranteeing data is collected from all clusters. In this setting, UAV trajectories were designed using heuristics, such as genetic algorithm and nearest-neighbor. The problem of minimizing the data collection time from time-constrained sensors was investigated in [6], where the

trajectory of a single UAV and radio resource allocation were optimized. The authors of [7] considered data collection using a UAV that transfers energy to IoT sensors to charge them for data sensing and transmitting. They optimized the UAV's trajectory aiming to minimize the average age of information (AoI) of the data collected from all SNs. Moreover, authors in [17] studied the joint optimization of SNs' wakeup schedule and a single UAV trajectory to minimize the total consumed energy by SNs, while guaranteeing data collection from all of them. You *et al.* used in [18] one UAV to collect data from SNs in an urban environment. They proposed a two-step data collection approach, where the UAV's trajectory is optimized offline given the SNs' location information, then further adjusted based on the instantaneous channel state information to maximize the data collection rate. In [19], the authors proposed an energy-efficient differential evolution based method that jointly optimizes the UAV's trajectory and IoT devices transmission schedule in order to ensure reliable data collection and 3D device positioning. Lin *et al.* presented in [20] an energy-efficient hierarchical data collection scheme in a wireless sensor network for agricultural monitoring. Following the introduction of compressed sampling based clustering, they proposed exact and greedy UAV path planning. Finally, authors of [21] formulated a mixed integer linear program to minimize the total traveled distance by UAVs when collecting data from clustered SNs. The optimal UAV trajectories were obtained and demonstrated to outperform the greedy benchmark, which directs the UAV to visit the closest unvisited CH at each step.

1.3.3 Online UAV path planning

Most of the previous works are based on complex optimization solutions, evolutionary algorithms, or heuristic approaches that require a priori knowledge of the IoT network's status, such as SNs/CHs locations, data deadlines, etc. However, in situations where information about the IoT system is not available, such as in rescue operations or in dynamic IoT networks where SNs are mounted, for instance, on moving vehicles or animals, previous path planning approaches become inefficient. In this context, reinforcement learning (RL), and in particular deep RL (DRL), comes out as a new strategy capable of providing intelligence to UAVs in order to make smart flying decisions and accomplish their mission. In DRL, an agent is trained by exploring unknown environments and exploiting received feedback in an online manner [22].

Yi *et al.* proposed in [23] a DQN based UAV trajectory that aims to minimize the weighted sum of the IoT data's AoI, while a similar method is presented by Tong *et al.* in [24] to jointly reduce the SNs' average AoI and packet drop

rate. Moreover, Bayerlein *et al.* demonstrated in [25] that their DDQN method combined with experience replay for UAV trajectory planning is efficiently generalized for different scenarios balancing between the data collection goal and flight time efficiency. Then, they extended their results to multiple UAVs in [26], where a multi-agent RL approach was proposed to control a swarm of cooperative UAVs. In [27], the authors proposed a green dueling DQN based data collection method to minimize UAV energy consumption while satisfying the delay constraints of different IoT data types. Hsu *et al.* investigated in [28] a UAV-based communication platform, where UAVs deliver data in the forward path and collect data from IoT devices in the backward path. A hybrid offline-online path planning method was proposed, where the offline solution pre-plans the backward trajectory independently from the other UAVs, and the online Q-learning solution guarantees that the path is collision-free. Finally, Bouhamed *et al.* proposed in [29] autonomous UAV-based data collection for delay-tolerant wireless sensor networks. The authors used deep deterministic gradient decent to determine the best obstacle-free path, while Q-learning was exploited for SNs' visits ordering such that data collection time is minimized.

In a nutshell, we present an objective and feature-based comparison of most relevant related works in Table 1.1 where (DCol) refers to data collection, (Vol.) refers to volume, (DC) refers to deployment costs (No. of CHs and/or UAVs), (G2G) refers to the ground-to-ground channel, and (G2A) refers to the ground-to-air channel

Table 1.1: Comparison of related works for UAV-based data collection in IoT networks

Papers	No. of UAVs	Focus		Objective / Performance metrics				Features		
		UAV path planning	SNs scheduling	DCol (Vol./Time)	Energy	DC	AoI	CHs	Deadlines	Channel model
Alfattani *et al.* [4]	Multiple	Offline	–	✓	–	✓	–	✓	–	G2G: Log-distance G2A: Probabilistic
Samir *et al.* [6]	Single	Offline	–	✓	–	–	–	–	✓	G2A: Rician
Hu *et al.* [7]	Single	Offline	–	✓	–	–	✓	–	✓	G2A: Free-space
Ghdiri *et al.*[30]	Multiple	Offline	–	✓	✓	✓	–	✓	✓	G2G: Log-distance G2A: Probabilistic
Zhan *et al.* [17]	Single	Offline	✓	✓	✓	–	–	–	–	G2A: Rician
You *et al.* [18]	Single	Hybrid offline-online	–	✓	–	–	–	–	–	G2A: Probabilistic
Wang *et al.* [19]	Single	Offline	✓	✓	✓	–	–	–	–	G2G: Rayleigh;Gaussian G2A: Log-distance
Lin *et al.* [20]	Single	Offline	–	✓	✓	–	–	✓	–	Not available
Albu-Salih *et al.* [21]	Multiple	Offline	–	✓	✓	–	–	✓	✓	Not available
Yi *et al.* [23]	Single	Online (RL)	–	✓	–	–	✓	–	✓	G2A: Free-space
Tong *et al.* [24]	Single	Online (RL)	–	✓	–	–	✓	–	✓	Not available
Bayerlein *et al.* [25]	Single	Online (RL)	–	✓	–	–	–	–	–	G2A: Log-normal shadowing
Bayerlein *et al.* [26]	Multiple	Online (RL)	–	✓	–	–	–	–	–	G2A: Log-normal shadowing
Liu *et al.* [27]	Single	Online (RL)	–	–	✓	–	–	–	–	Not available
Hsu *et al.* [28]	Multiple	Hybrid offline-online (RL)	–	✓	–	–	–	–	–	G2A: Log-distance
Bouhamed *et al.* [29]	Single	Online (RL)	✓	✓	–	–	–	–	–	G2A: Probabilistic
This Work	Multiple	Offline and online (RL)	–	✓	✓	✓	–	✓	✓	G2G: Log-distance G2A: Probabilistic

1.4 Main contributions

Key contributions of this project can be summarized as follows:

- Unlike state-of-the-art studies with simplified assumptions for UAV-enabled data collection, such as using a single UAV, impractical communication models, unlimited energy, or uncharacterized data, this work proposes a full framework for multi-UAV data collection in a time-constrained IoT network, where several real environment constraints are taken into account, e.g., data deadlines, UAV characteristics, and realistic channel models.
- We formulate the problem of maximizing the amount of collected data from CHs while using the lowest UAV energy, through the optimization of SNs clustering, number and locations of CHs, and number and trajectories of UAVs.
- To solve the formulated NP-hard problem, we propose a two-step solution. First, clustering and minimum number of CHs and their locations are determined through the proposal of a novel K-means based approach. Then, the number of collecting UAVs and their trajectories are optimized for two network scenarios. In the first scenario, we assume a prior and full knowledge of the system's status, including data deadlines and CHs locations. Thus, UAV paths are optimized offline using near-optimal meta-heuristic approaches. In the second scenario, we assume that no information about the system is available. Hence, we propose online path planning based on DRL algorithms.
- Through simulations, we show the efficacy of our approaches in reducing the deployment costs in terms of number of CHs and UAVs, and maximizing data collection using the lowest consumed UAV energy.

Conclusion

An introduction gives an overview of this research project, including the background related to this project, the motivation and the objectives to reach. Before elaborating the solution, we present an insight to previous developed research as well as our contributions. The next chapter provides an overview of some important theoretical concepts, certain selected approaches and algorithms in order to solve the aforementioned problems.

CHAPTER 2

THEORETICAL BACKGROUND

Introduction

The vision of future wireless technologies with numerous smart applications and services has emerged with the Internet of Things (IoT). IoT sensor networks are a key enabler for novel applications, including smart cities, connected cars, smart grids, and intelligent transportation. However, a major concern for the myriad of connected devices is the need to gather a very large amount of data generated frequently. This chapter provides background on the concepts and methods used in this project. Specifically, it aims to provide the detailed definition and the necessary theoretical background on IoT, UAVs, Vehicle Routing Problem and Reinforcement Learning, in order to understand the terms and discussions that will be presented in the upcoming chapters.

2.1 Internet of Things

2.1.1 General overview

The term Internet of Things (IoT) is known to everyone, but it is hard to find a single universally accepted definition. Generally, IoT denotes a trend where network connectivity and computing capability extend to things or objects of day-to-day life that will allow these devices to generate, exchange and consume data through the Internet.

Over the last few years, the vision of Internet of Things has been constantly expanding in every aspect of our life. While IoT does not consider a particular communication technology, wireless communication technologies are considered as an essential pillar to enabling promising IoT applications, connecting millions of devices and people to change our lives in countless ways. In particular, wireless sensor networks (WSN) have been gaining momentum in the last few years due to their flexibility in solving numerous problems in different application domains.

WSN can generally be described as a set of interconnected sensor nodes (SNs) that communicate through a wireless channel to capture certain data about the surrounding environment. The tremendous growth in wireless communications have made it possible to develop tiny, low-cost, multi-functional sensors. SNs could be static or mobile deployed over a dense area in order to sense, communicate, collect and forward data to a control center, where data storage and processing are performed.

The large-scale implementation of IoT devices promises to transform many aspects of our lives but raises many issues, namely the demand for a higher throughput, larger capacity, and lower latency for users. Due to the speed, massive capacity and super low latency of the fifth generation (5G), there is a belief that the desired requirements will be fulfilled. Projections for the impact of IoT on the Internet and economy are impressive, with some anticipating as many as 100 billion connected IoT devices and a global economic impact of more than $11 trillion by 2025 [31]. Accordingly, 5G-enabled IoT is expected to radically change the quality of life and spur innovation across many fields, including ecology, transport, entertainment, health, and security.

2.1.2 Applications of IoT

IoT based applications mainly consist of the ability to enable communications between an infinite amount of devices, which have significantly changed perspectives for various fields of application, as depicted in Figure 2.1 below:

Smart grid [32] - The next-generation electricity supply network that uses digital communications technology to reliably and efficiently control energy system. It ensures economically-efficient, sustainable power system with low losses and high levels of quality and security of supply and safety.

Smart cities [33] - The utilization of IoT devices such as connected sensors, lights, and meters to collect and analyze data in order to improve infrastructure, public utilities and services.

Medical and healthcare [34] - The systematic application mode that connects healthcare services to the IoT systems, i.e. medical device monitoring, health team coordination, optimizing workflow operations, while out-patient focused solutions include patient monitoring, assisted living, elderly care, and pain medication management.

Environment monitoring and public safety [35] - The process of observing and studying environmental conditions and trends such as weather conditions, endangered species protection, water quality monitoring, and various other parameters that are associated directly or indirectly to our environment.

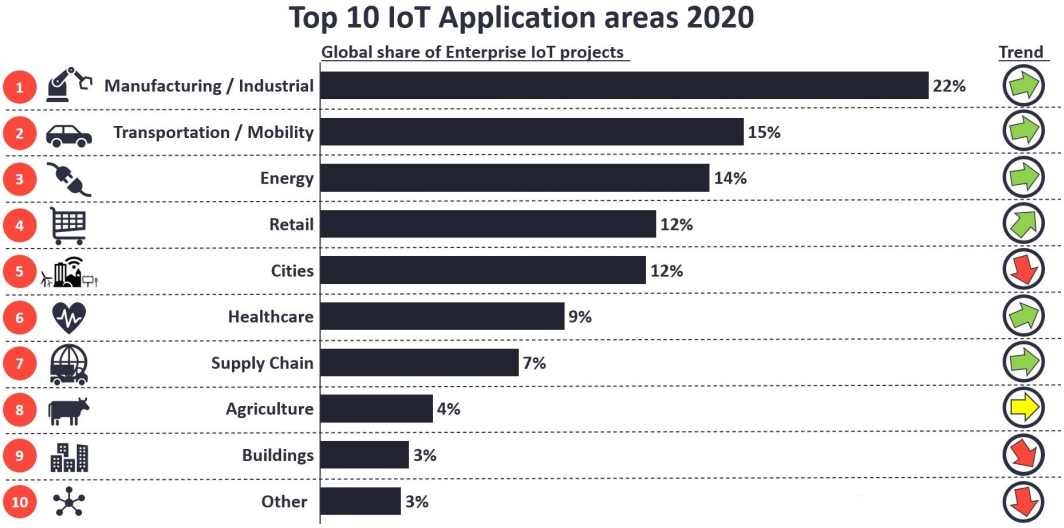


Figure 2.1: Top 10 IoT applications in 2020 [36]

The study of IoT applications improves the understanding and enhancement of IoT technology, and thus, potential new applications can be identified and developed to meet future technology and market trends.

2.1.3 Challenges in IoT

Even though the current IoT enabling technologies have greatly improved in the recent years, there are still numerous issues and challenges that need to be considered and addressed in order to unleash their full potential. Considering the wide variety of IoT applications, some of the challenges are listed below:

Privacy and Security It is one of the biggest challenges in IoT. As we increasingly connect devices to the Internet, new opportunities to exploit potential security vulnerabilities grow. Thus, IoT devices could be a target for cyberattacks, which can expose user data to theft and malicious manipulation.

Reliability When the power goes out, or the local Internet service provider is experiencing an outage, entire systems go offline. Moreover, natural disasters or emergencies can also affect data centers. That can mean many different things in the world of IoT. Sometimes entire devices or systems become unusable. Other times, they might still operate but at a reduced capacity. Worse yet, the data collection and reporting processes can be totally interrupted.

Data collection and management The procedure for processing, analyzing and managing data is tremendously challenging because of the heterogeneous nature of IoT, and the large scale of data collected, particularly in this era of Big Data.

Currently, most systems rely on centralized data offloading architectures that carry out computationally intensive tasks on cloud platforms. Nevertheless, there is a constant concern about conventional cloud architectures not being effective in terms of transferring the massive volumes of generated and consumed IoT data, support the accompanying computational load, and simultaneously meet timing constraints. Given these massive amounts of data, IoT systems are struggling with how much and exactly what data to collect and store. In addition, energy consumption continues to remain a barrier challenge to transmit data for long ranges.

2.2 Unmanned Aerial Vehicle

Unmanned aerial vehicles (UAVs), widely known as drones, are aircraft piloted by remote control or embedded computer programs without an onboard human pilot or passengers. Although drones are mostly used in military applications nowadays, they have also emerged in consumer market for either professional or recreational uses.

The main goal of the UAV is to fulfill a mission, but planning and executing it in terms of prescribing the path or trajectory planning are challenging problems that require careful consideration. It is hard to achieve a unique classification for UAVs since it differs between countries. The classification depends on several parameters such as altitude range, endurance and weight. Also, UAVs support a wide range of applications including military, scientific, or commercial applications. A comprehensive classification of UAVs demonstrating both the wide variety of UAV systems and capabilities, as well as the multiple dimensions of differentiation, is presented in Table 2.1 below.

UAVs can also be categorized, based on type, into fixed-wing and rotary-wing

Table 2.1: UAV categorization for differentiation of existing systems

	Mass (kg)	Range (km)	Flight alt. (m)	Endurance (h)
Micro	<5	<10	100	1
Mini	<20	<10	250	<2
Close range	25-150	10-30	3000	2-4
Short range	50-250	30-70	3000	3-6
Medium range	150-500	70-200	5000	6-10

UAVs. Compared to rotary-wing UAVs, fixed-wing UAVs, such as small aircrafts, are heavier, faster, and they need to move forward in order to remain aloft. In contrast, rotary-wing UAVs, such as quadrotor drones, can hover and remain stationary over a given area [37].

In Figure 2.2, we provide an overview on the different types of UAVs, their functions, and capabilities. We note that the flight time of a UAV depends on several factors such as energy source (e.g., battery, fuel, etc.,), type, weight, speed, and trajectory.

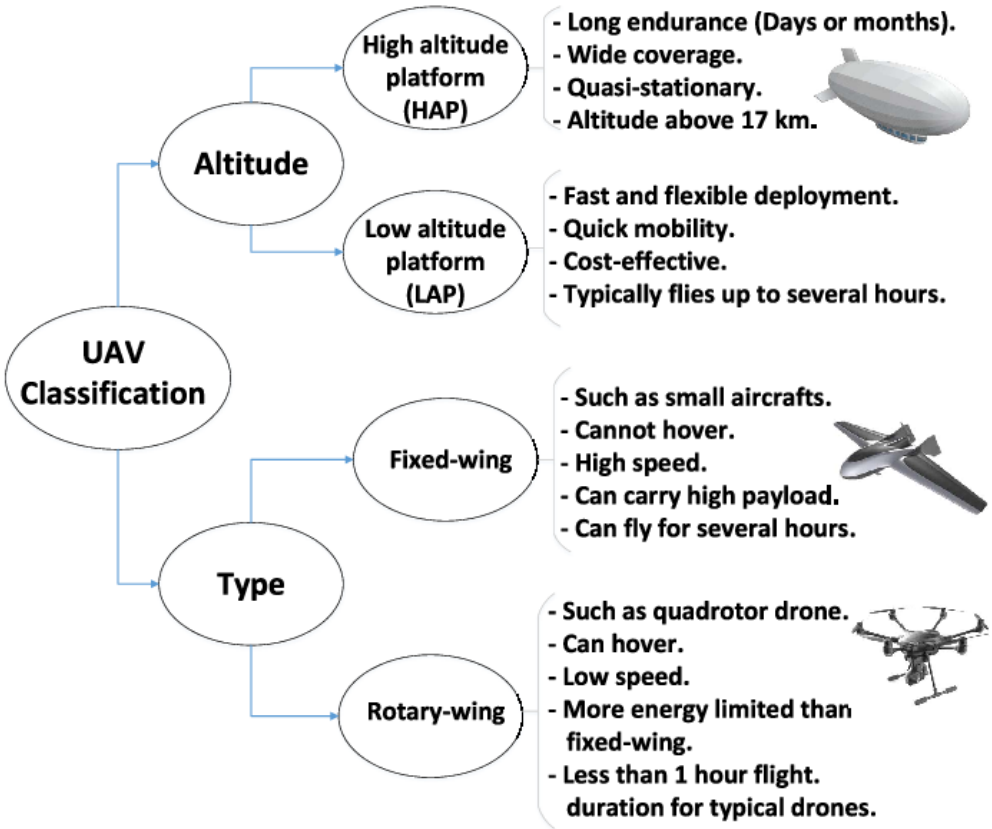


Figure 2.2: UAV classification [37]

2.3 IoT clustering techniques

Clustering analysis has been an emerging research issue in data mining due its variety of applications. With the advent of many data clustering algorithms in the recent years and their extensive use in a wide variety of applications, including image processing, computational biology, mobile communication, medicine and economics, these algorithms gained a lot of popularity.

As SNs are resource constrained nodes, they have small size battery, limited processing power and low communication range. These constrains, along with the massive amount of data, require novel energy conservation strategies. Out of several proposed solutions, clustering is one of the distinguished options.

IoT clustering is the task of gathering SNs into a number of groups based on their similarity. The similarity or dissimilarity between two data objects is typically measured as the distance between the multi-dimensional feature vectors that represent the objects. The resulting groups having similar data points are called clusters [38]. Clustering is the unsupervised classification of patterns into groups, i.e. it only requires datasets having data points without labels.

IoT clustering offers certain advantages such as scalability, robustness and energy efficiency. Any clustering algorithm needs to perform not only cluster formation but also a cluster head (CH) selection among nodes within each cluster. In our approach, sensors can transmit data to a cluster head, which aggregates data from all sensors in its cluster. CHs are responsible for the administration of nodes within their cluster and periodically collect data from them. Eventually, the sensor nodes are able to communicate with the collecting control center through the CHs, which reduces overall communication costs and hence reduces energy consumption to prolong the network's lifetime.

The major problem with the data clustering algorithms is that they cannot be standardized. In other words, a certain algorithm may give good results with one type of datasets but may fail or provide poor results on other datasets. To find the most suitable clustering technique, we compare two different clustering categories, namely partitioning and hierarchical methods.

2.3.1 Partitioning methods

These methods partition data into k subsets, where each subset is a cluster. Each object must belong to exactly one cluster, and no cluster can be empty. There are various objective functions that are optimized under this type of methods, with the most common being the sum of squared distances between every data point

and its cluster centroid. Clusters found with this type of methods are spherical.

Example: K-means

K-means clustering is one of the simplest and frequently used unsupervised learning algorithms, especially in data mining and statistics [39]. The k-means algorithm works by randomly initializing k random points in the dataset, called the cluster centroids. The number k is chosen by hand or derived using an evaluation method. It then proceeds repetitively with two steps: Assignment and centroid repositioning. In the cluster assignment step, the algorithm iterates through each of the examples in the given dataset and assigns each example to one of the initialized centroids based on the closest distance. This is repeated for each data point until every example is assigned to a cluster. In the second step, the algorithm computes the average distance for each data point that is assigned to a specific cluster. The centroid is then moved to the calculated mean location. This step is repeated for all k clusters. The algorithm iterates until the cluster centroids no longer move, meaning the k-means algorithm has converged to k clusters. The result of the algorithm can vary due to the random initialization of the centroids. When applied in practice, the k-means algorithm is deployed multiple times with different initializations of the centroids and varying numbers of centroids k in order to achieve a practical clustering result.

2.3.2 Hierarchical methods

There are two main types of hierarchical methods: Agglomerative and divisive [40]. Agglomerative clustering starts with each data point forming its own singleton cluster, and iteratively merges pairs of clusters that are closest to one another, until one cluster is formed. The divisive approach goes in the opposite direction. That is, it starts with a single cluster with all the data points in it, and iteratively divides it into two clusters that are furthest apart from one another. There are several methods to calculate the distance between two clusters, such as single, complete, or average linkage, and Ward's method, which minimizes the increase in total within-cluster sum of squared distances when two clusters are merged. Cluster structure found using hierarchical clustering can be represented graphically by a dendrogram.

Example: Hierarchical Agglomerative Clustering

The agglomerative clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. To perform Hierarchical Agglomerative Clustering (HAC) [41] on a set of n data points, a symmetric

$n \times n$ distance matrix consisting of pair-wise distances between the data points is generated. First, assign each data point to a separate cluster to obtain n clusters, each of which contains one data point (a singleton). Next, pairs of the closest clusters are successively merged until all clusters have been merged into one big cluster containing all objects. In general, when employing hierarchical clustering to build a single tree, the user does not need to specify the desired number of clusters, k. However, to obtain a clustering result with a specific number of clusters, the user does need to provide the value of k, so that the algorithm can cut the tree at a certain level where there are exactly k sub-trees under this level. Figure 2.3 shows how to obtain four clusters from the dendrogram.

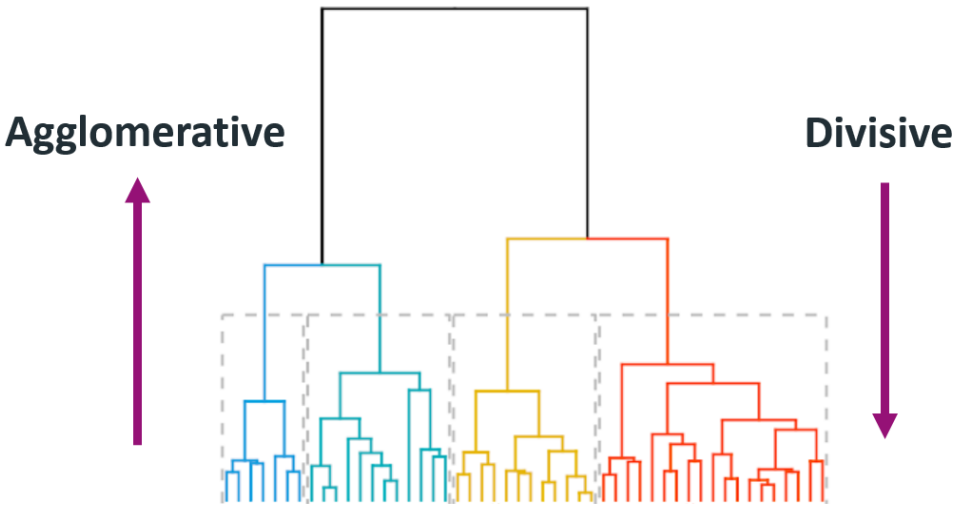


Figure 2.3: A dendrogram representing the clustering technique of hierarchical clustering algorithm

2.4 The Vehicle Routing Problem

2.4.1 Definition and background

The vehicle routing problem (VRP) is a complex combinatorial optimization problem that can be seen as a merge of two well-known problems: Traveling Salesman Problems (TSPs) and Bin Packing Problems (BRPs). Proposed by Dantzig and Ramser in 1959 [42], VRP is a generic name given to a whole class of NP-hard problems seeking to service a fleet of vehicles and have many practical applications, especially in transportation and distribution logistics. The fundamental objectives are to find the minimal number of vehicles, the minimal travel time or the minimal costs of the travelled routes for a fleet of vehicles originating and terminating at a depot.

In practice, the basic formulation of the VRP problem is augmented by constraints such as vehicle limitations, cost controls, time windows and resource limitations concerning the loading process at the depot. Beyond the classical VRP, some other variants have also been studied. For instance, only capacity restrictions for vehicles are considered in Capacited VRP (CVRP), and the challenge is to save costs by transporting more goods in a single trip without exceeding the vehicle's capacity. Moreover, in Capacitated Vehicle Routing Problem with Time Windows (CVRPTW), which is a generalization of the CVRP, the vehicles must comply with a constraint of time windows associated with each customer in which a vehicle has to reach a customer within a prioritized time frame. The vehicle should serve the customer during specific time windows that dictates the need to schedule rides.

2.4.2 Solving VRP

A large variety of VRP solution strategies are available in literature. These range from exact methods to the approximate solution methods. While exact methods provide the optimal i.e., the best solutions, approximate methods generally yield near-optimal results.

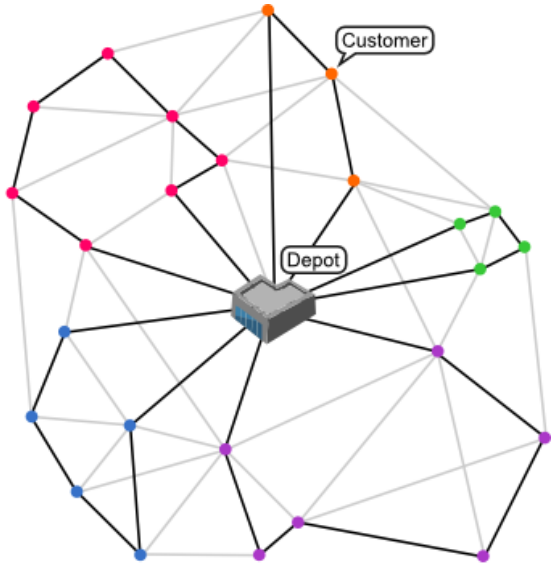


Figure 2.4: An example of a VRP solution

2.4.2.1 Exact approaches

Classical approaches to exactly solve a combinatorial optimization problem are Branch & Bound and Dynamic Programming. They are divide-and-conquer methods, since both solve a problem by combining the solutions to its subproblems. Generally, exact methods are suitable for only small sized problems.

2.4.2.2 Approximation approaches

Approximation methods find suboptimal solutions. Providing an approximation guarantees the quality of the found solution. These methods are applied for approaching intractable combinatorial optimization problems, since they are, in a sense, the best option for a solution quality guarantee. Due to the computational complexity of hard combinatorial problems, especially in the presence of large scale problem instances, in the last decades there has been an extensive research effort devoted to the development of:

Heuristics
Heuristics are techniques designed for solving a problem quicker than classic methods when these are too slow, or for finding an approximate solution when classic methods fail to find any optimal solution. This is accomplished by trading optimality, completeness, accuracy, or precision for speed. In a way, it can be considered as a shortcut. The effectiveness of a heuristic method depends upon its ability to adapt to a particular realization, avoid entrapment at local optima, and exploit the basic structure of the problem. Building on these notions, various heuristic search techniques have been developed that have demonstrably improved our ability to obtain good solutions to difficult combinatorial optimization problems.

Meta-heuristics
Meta-heuristics are a class of methods commonly applied to suboptimally solve computationally intractable combinatorial optimization problems. The term meta-heuristic derives from the composition of two Greek words: Meta and heuriskein. The suffix 'meta' means 'beyond', 'in an upper level', while 'heuriskein' means 'to find'. In fact, meta-heuristics are a family of algorithms that try to combine basic heuristic methods in higher level frameworks aiming to efficiently explore the set of feasible solutions of a given combinatorial problem. The most promising techniques include, but are not limited to, simulated annealing, Tabu search, evolutionary algorithms like genetic algorithms, and ant colony optimization.

Examples of meta-heuristic algorithms:

Guided local search (GLS) [43]: uses a meta-heuristic based on penalties to continuously modify the penalty terms associated with an augmented objective function to escape from local minima for efficiently solving a wide range of constrained optimization problems.

Tabu search [44]: is an interesting meta-heuristic, which aims to model the

human memory process through the use of a Tabu list, usually of a fixed length, to prohibit most recent moves possibly leading to some previously visited local optima.

2.5 Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning along with supervised and unsupervised learning. Supervised learning has the supervisor indicating the correct behavior in a certain generalized situation, whereas, unsupervised learning finds hidden structures in unlabeled data. In contrast with other methods, RL is an agent-based algorithm in which the agent learns to perform an optimal set of actions through interaction and experience. The interactive goal-directed learner is able to operate in an uncertain setup, make decisions despite uncertainty and predict future events. The agent interacts with its dynamic environment in a trial and error manner in order to maximize the rewards it receives based on the actions it takes, illustrated in figure 2.5.

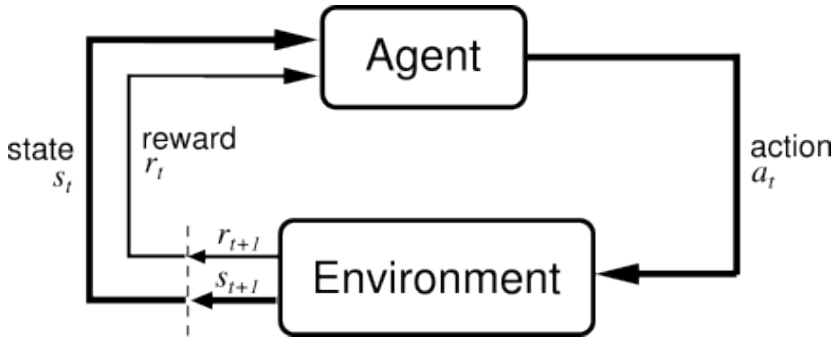


Figure 2.5: The agent-environment interaction loop

RL employs a number of terms of art to express its concepts. The terms interlock, that is, they are used to define each other. So before we dive into RL we need to introduce some basic terminology:

The agent is the entity that acts in the environment and tries to maximize its long-term sum of rewards. Firstly, it needs to be able to manipulate the environment which is often manipulating itself within the environment. Secondly, it needs a way to perceive the effects of its manipulations. Thirdly, it needs to use the obtained reward to approximate the expected sum of rewards that can be obtained from the different states.

The environment is the space where the agent lives, acts and performs its task. The environment is assumed to contain everything pertinent to the problem. When the agent picks an action, the environment moves the agent to the next state based on its current state and the transition probabilities.

State (s) is a concrete and immediate representation in which the agent finds itself. The state should contain enough information to be able to make a decision.

Observation is a partial description of a state. When the agent is able to observe the complete state of the environment, we say that the environment is fully observed. When the agent can only see a partial observation, we say that the environment is partially observed.

Action (a) is a move that agent can make. The set of all valid actions in a given environment is often called the action space. Some environments, have discrete action spaces, where only a finite number of moves are available to the agent. Other environments, have continuous action spaces.

Transition model This is a probabilistic model that maps the probability of ending up in state s' given the agent is at s and takes action a. It is expressed by $\mathcal{P}_a(\mathbf{s}, \mathbf{s}')$.

Reward (r) The reward is designed such that maximizing its sum leads to the best performance of the task. The reward function is represented by $\mathcal{R}_a(\mathbf{s}, \mathbf{s}')$ which expresses the expected reward at the next time step given action a, current state s and expected next state s'.

Episode is the whole sequence of state, action and reward which ends with a terminal state or a predefined length of actions. It is noteworthy that different episodes are completely independent of one another.

Policy π is a rule used by an agent to decide what actions to take. It can be deterministic or stochastic.

2.5.1 Markov decision process

In order to mathematically formalize any decision-making effort, a common framework is required to compare real world tasks or theoretical problems. Markov decision process (MDP) offers a standard formalism for describing multi-state decision making in probabilistic environment.

More precisely, a MDP is a discrete time stochastic control process and must satisfy the Markov property which is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where:

- $\mathcal{S}$ is a set of states observable by the agent that can define the environment in a particular instant.
- $\mathcal{A}$ is a set of actions available to the agent to perform.
- $\mathcal{P}$ is the state transition probability function describing the probability of transition between states s and s' when performing an action a.
- $\mathcal{R}$ is the reward function determining the immediate reward of performing an action a from a state s, resulting in state s'.
- $\gamma \in [0, 1]$ is called the discount factor where lower values place more emphasis on immediate rewards.

This definition of MDP allows us to model episodic tasks where the goal is to take the agent from some starting state to a terminal state. In these cases, one can distinguish between fixed horizon tasks in which each episode consists of a fixed number of steps, or indefinite horizon tasks in which each episode ends when we reach a specific terminal state.

2.5.2 Functions to improve behaviour

Behaviour on the part of our agent was not yet mathematically specified. An agent is expected to progressively collect more rewards within each episode, thus actively learning by reinforcement. Each state is followed by an action, which leads to another state and corresponding reward. The agent's behaviour becomes a simple question: "What action should I take for each state?".

2.5.2.1 Policy

The behavior of the agent in an MDP can be defined as a probability distribution $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ called a policy, which given $s \in \mathcal{S}$ and $a \in \mathcal{A}$, represents the probability of selecting a as next action from s. An agent that uses this probability distribution to select its next action when in a given state is said to be following the policy.

A common problem when defining policies is the exploration-exploitation dilemma. An agent following a policy may end up observing the same trajectories in all episodes (e.g. when following a deterministic policy in a deterministic MDP), but there may be cases in which a better behavior could be had if the agent explored other states instead of simply exploiting its knowledge. It is there-

fore common to add a probabilistic element to policies, in order to explicitly control the exploration degree of the agent. Common techniques to control the exploration-exploitation trade-off are:

- ε-greedy policies, in which actions are selected using a given policy with probability $1 - \varepsilon$, and randomly the rest of the time.
- softmax action selection, that improves on ε-greedy policies by reducing the number of times a suboptimal action is randomly selected. To do so, a probability distribution, commonly a Boltzmann distribution, dependent on the expected return from the successor states is used.

2.5.2.2 Value Functions

Starting from the concept of policy, we can now introduce a function that evaluates how good it is for an agent following a policy π to be in a given state. This evaluation is expressed in terms of the expected return, i.e. the expected discounted sum of future rewards collected by an agent starting from a state while following π, and the function that computes it is called the state-value function for policy π.

Formally, the value function associated to a policy π is a function $V^\pi : \mathcal{S} \to \mathcal{R}$ defined as:

$$\begin{aligned} V^\pi(s) &= E_\pi[R_t|s_t = s] \\ &= E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s] \end{aligned} \tag{2.1}$$

where $E_\pi[\cdot]$ is the expected value given that the agent follows policy π, and t is any time step of an episode $[s_0, ..., s_t, ..., s_n]$ where $s_t \in \mathcal{S}, \forall t = 0, ..., n$.

Similarly, we can also introduce a function that evaluates the goodness of taking a specific action in a given state, namely the expected reward obtained by taking an action $a \in \mathcal{A}$ in a state $s \in \mathcal{S}$ and then following policy π. We call this function the action-value function for policy π denoted $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$, and defined as:

$$\begin{aligned} Q^\pi(s, a) &= E_\pi[R_t|s_t = s, a_t = a] \\ &= E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a] \end{aligned} \tag{2.2}$$

The majority of reinforcement learning algorithms is based on computing or estimating the value functions, which can then be used to control the behavior of the

agent. We also note a fundamental property of the value functions, which satisfy particular recursive relationships like the following Bellman equation for V^π:

$$\begin{aligned} V^\pi(s) &= E_\pi[R_t|s_t = s] \\ &= E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s] \\ &= E_\pi[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2}|s_t = s] \\ &= \sum_{a\in A} \pi(s,a) \sum_{s'\in S} P(s,a,s') \Big[R(s,a) + \\ &\quad + \gamma E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2}|s_{t+1} = s']\Big] \\ &= \sum_{a\in A} \pi(s,a) \sum_{s'\in S} P(s,a,s')[R(s,a) + \gamma V^\pi(s')] \end{aligned} \tag{2.3}$$

Intuitively, Equation (2.3) decomposes the state-value function as the sum of the immediate reward collected from a state s to a successor state s', and the value of s' itself; by considering the transition model of the MDP and the policy being followed, we see that the Bellman equation simply averages the expected return over all the possible (s, a, r, s') transitions, by taking into account the probability that these transitions occur.

2.5.2.3 Optimal Value Functions

In general terms, solving a reinforcement learning task consists in finding a policy that yields a sufficiently high expected return. In the case of MDPs with discrete state and actions sets, it is possible to define the concept of optimal policy as the policy that maximizes the expected return collected by the agent in an episode.

We start by noticing that state-value functions define a partial ordering over policies as follows:

$$\pi \geq \pi' \iff V^\pi(s) \geq V^{\pi'}(s), \forall s \in \mathcal{S}$$

From this, the optimal policy π^* of an MDP is a policy that is better or equal than all other policies in the policy space. It has also been proven that among all optimal policies for an MDP, there is always a deterministic one.

The state-value function associated to π^* is called the optimal state-value function, denoted V^* and defined as:

$$V^*(s) = \max_\pi V^\pi(s), \forall s \in \mathcal{S} \tag{2.4}$$

As we did when introducing the value functions, given an optimal policy for the MDP it is also possible to define the optimal action-value function denoted Q^*:

$$\begin{aligned} Q^*(s,a) &= \max_\pi Q^\pi(s,a) \\ &= E[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a] \end{aligned} \tag{2.5}$$

Notice that Equation (2.5) in this definition highlights the relation between Q^* and V^*. Since V^* and Q^* are value functions of an MDP, they must satisfy the same type of recursive relations that we described in Equation (2.3), in this case called the Bellman optimality equations.

The Bellman optimality equation for V^* expresses the fact that the value of a state associated to an optimal policy must be the expected return of the best action that the agent can take in that state:

$$\begin{aligned} V^*(s) &= \max_a Q^*(s,a) \\ &= \max_a E_{\pi^*}[R_t | s_t = s, a_t = a] \\ &= \max_a E_{\pi^*}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a] \\ &= \max_a E_{\pi^*}[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a] \\ &= \max_a \sum_{s' \in S} P(s,a,s')[R(s,a) + \gamma V^*(s')] \end{aligned} \tag{2.6}$$

The Bellman optimality equation for Q^* is again obtained from the definition as:

$$\begin{aligned} Q^*(s,a) &= E[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') || s_t = s, a_t = a] \\ &= \sum_{s'} P(s,a,s')[R(s,a) + \gamma \max_{a'} Q^*(s', a')] \end{aligned} \tag{2.7}$$

Notice that both Bellman optimality equations have a unique solution independent of the policy. If the dynamics of the environment are fully known, it is possible to solve the system of equations associated to the value functions (i.e. one equation for each state in $\mathcal{S}$) and get an exact value for V^* and Q^* in each state.

2.5.3 Tabular solution methods

2.5.3.1 Dynamic Programming

The use of dynamic programming (DP) techniques to solve reinforcement learning problems is based on recursively applying some form of the Bellman equation, starting from an initial policy or value function until convergence to an opti-

mal policy or value function. In this class of algorithms, we identify two main approaches: policy iteration and value iteration.

Policy Iteration

Policy iteration is based on the following theorem:

Theorem 2.1 (Policy improvement theorem)**.** *Let π and π' be a pair of deterministic policies such that*

$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s), \forall s \in S$$

Then, $\pi' \geq \pi$, i.e.

$$V^{\pi'}(s) \geq V^{\pi}(s), \forall s \in S$$

This approach works by iteratively computing the value function associated to the current policy, and then improving that policy by making it act greedily with respect to the value function, such that:

$$\pi'(s) = \arg\max_{a \in A} Q^{\pi}(s, a) \tag{2.8}$$

For Theorem 2.1, the expected return of the policy is thus improved because:

$$Q^{\pi}(s, \pi'(s)) = \max_{a \in A} Q^{\pi}(s, a) \geq Q^{\pi}(s, \pi(s)) = V^{\pi}(s) \tag{2.9}$$

This continuous improvement is applied until Inequality (2.9) becomes an equality, i.e. until the improved policy satisfies the Bellman optimality equation (2.6). Since the algorithm gives no assurances on the number of updates required for convergence, some stopping conditions are usually introduced to end the process when the new value function does not change substantially after the update or a certain threshold number of iterations has been reached.

Value Iteration

Based on a similar idea, the value iteration approach starts from an arbitrary value function, and then applies a contraction operator which iterates over sequentially better value functions without actually computing the associated greedy policy. The contraction operator which ensures convergence is the Bellman optimality backup:

$$V_{k+1}(s) \leftarrow \max_{a} \sum_{s'} P(s, a, s')[R(s, a) + \gamma V(s')] \tag{2.10}$$

As with policy iteration, convergence is ensured without guarantees on the number of steps, and therefore it is usual to terminate the iteration according to some stopping condition.

2.5.3.2 Temporal Difference Learning

Temporal Difference (TD) learning is an approach to RL that uses concepts from both dynamic programming and Monte Carlo techniques. TD is a model-free approach that uses experience to update an estimate of the value function by using a previous estimate.

In general, TD methods have several advantages as they allow for an online (i.e. they don't require full episode trajectories to work), bootstrapped, model-free estimate, which is more suitable for problems with long or even infinite time horizons. Moreover, TD is less susceptible to errors and to the effects of exploratory actions, and in general provides a more stable learning.

The two principal control algorithms in the TD family, one said to be on-policy (i.e. methods that attempt to evaluate and improve the same policy that they use to make decisions) and the other off-policy (i.e. methods with no relations between the estimated policy and the policy used to collect experience).

SARSA

As usual in on-policy approaches, SARSA works by estimating the value $Q^\pi(s, a)$ for a current behavior policy π which is used to collect sample transitions from the environment. The policy is updated towards greediness with respect to the estimated action-value after each transition (s, a, r, s', a'), and the action-value is in turn updated step-wise with the following rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{2.11}$$

The training procedure of SARSA can be summarized with Algorithm 1.

Convergence of the SARSA method is guaranteed by the dependence of π on the action-value function, as long as all state-action pairs are visited an infinite number of times and the policy converges in the limit to the greedy policy (e.g. a time-dependent ε-greedy policy with $\varepsilon = 1/t$).

Q-learning

Introduced by Watkins in 1992 [45] is one of the most important breakthroughs in reinforcement learning. Q-learning is an off-policy temporal difference method that approximates the optimal action-value function independently of the policy

Algorithm 1 SARSA

```
Initialize Q(s, a) arbitrarily
Initialize π as some function of Q (e.g. greedy)
repeat
    Initialize s
    Choose a from s using π
    repeat
        Take action a, observe r, s′
        Choose a′ from s′ using π
        Update Q(s, a) using Rule (2.11)
        if π is time-variant then
            Update π towards greediness
        end if
        s ← s′; a ← a′
    until s is terminal or Q did not change
until training ended or Q did not change
```

being used to collect experiences. This simple, yet powerful idea guarantees convergence to the optimal value function as long as all state-action pairs are continuously visited during training.

The update rule for the TD step in Q-learning is the following:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \qquad (2.12)$$

As we did for SARSA, a procedural description of the Q-learning algorithm is provided in Algorithm 2.

Algorithm 2 Q-Learning

```
Initialize Q(s, a) and π arbitrarily
repeat
    Initialize s
    repeat
        Choose a from s′ using π
        Take action a, observe r, s′
        Update Q(s, a) using Rule (2.12)
        s ← s′
    until s is terminal or Q did not change
until training ended or Q did not change
```

2.5.4 Approximate solution methods

The integration of RL and neural networks has a long history. But, the recent and exciting achievements of DL have caused a sort of RL renaissance, with DRL

algorithms outperforming classic RL techniques on high-dimensional problems. This is due to the learning of low-dimensional feature representations and the powerful function approximation properties of neural networks.

2.5.4.1 Value-based Methods

Deep Q-Learning

In 2013, Mnih et al. [22] introduced the deep Q-learning (DQN) algorithm, which basically ignited the field of DRL. The important contributions of DQN consisted in providing an end-to-end framework to train an agent on the Atari environments starting from the pixel-level representation of the states, with a deep CNN used to estimate the Q function and apply greedy control. The authors were able to reuse the same architecture to solve many different games without the need for hyperparameter tuning, which proved the effectiveness of the method.

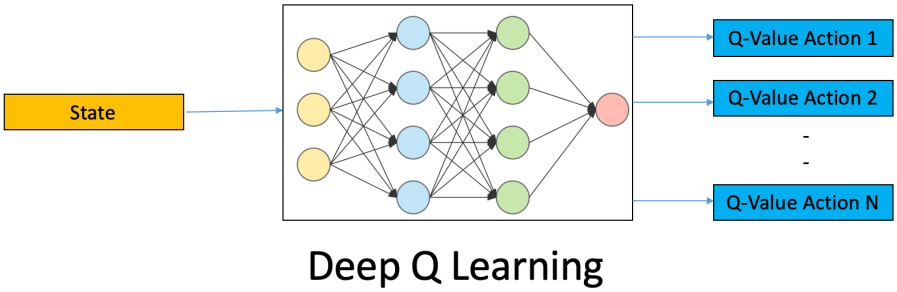


Figure 2.6: Structure of the Deep Q-Network [22]

The key idea of DQN is to embed the update step of Q-learning into the loss used for stochastic gradient descent (SGD) to train the deep CNN, resulting in the following gradient update:

$$\frac{\partial L}{\partial W_i^{old}} = E[(r + \gamma \max_{a'} Q(s', a'; \theta') - Q(s, a)) \frac{\partial Q(s, a; \theta)}{\partial W_i^{old}}] \tag{2.13}$$

where θ, θ' indicate two different sets of parameters for the CNN, which are respectively called the online network (θ) to select the action for the collection of samples, and the target network (θ') to produce the update targets. The online network is continuously updated during training, whereas the target network is kept fixed for longer time intervals in order to stabilize the online estimate. Moreover, a sampling procedure called experience replay [46] is used to stabilize training. This consists in keeping a variable training set of transitions collected with increasingly better policies, from which training samples are randomly selected. The full training procedure of DQN is reported in Algorithm 3.

Algorithm 3 Deep Q-Learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with two random sets of weights θ, θ'
for $episode = 1, M$ **do**
  for $t = 1, T$ **do**
    Select a random action a_t with probability ε
    Otherwise, select $a_t = \arg\max_a Q(s_t, a; \theta)$
    Execute action a_t, collect reward r_{t+1} and observe next state s_{t+1}
    Store the transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $\mathcal{D}$
    Sample mini-batch of transitions $(s_j, a_j, r_{j+1}, s_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_{j+1}, & \text{if } s_{j+1} \text{ is terminal} \\ r_{j+1} + \gamma \max_{a'} Q(s_{j+1}, a'; \theta'), & \text{otherwise} \end{cases}$
    Perform a gradient descent step using targets y_j with respect to the online parameters θ
    Every C steps, set $\theta' \leftarrow \theta$
  end for
end for

Double Deep Q-Learning

From DQN, many improvements have been proposed in the literature. Van Hasselt et al. (2016) [47] proposed *Double DQN* (DDQN) to solve an over-estimation issue typical of Q-learning, due to the use of the maximum action value as an approximation for the maximum expected action value (see Equation (2.12)). Double Q-learning, a learning algorithm that keeps two separate estimates of the action-value function Q^A and Q^B, and uses one to update the other as follows:

$$Q^A(s,a) \leftarrow Q^A(s,a) + \alpha[r + \gamma Q^B(s', \arg\max_a Q^A(s',a)) - Q^A(s,a)] \quad (2.14)$$

The idea is to train independently two value networks: one will be used to find the greedy action (the action with the maximal Q-value), the other to estimate the Q-value itself. Even if the first network choose an over-estimated action as the greedy action, the other might provide a less over-estimated value for it, solving the problem.

DDQN uses a similar approach to limit over-estimation in DQN by using two value networks, the trained network and the target network. Instead of using the target network to both select the greedy action in the next state and estimate its Q-value, here the trained network is used to select the greedy action while the target network only estimates its Q-value. This is achieved with a small change

in the computation of the update targets:

$$y_j = \begin{cases} r_{j+1}, & \text{if } s_{j+1} \text{ is terminal} \\ r_{j+1} + \gamma Q(s_{j+1}, \arg\max_a Q(s_{j+1}, a; \theta); \theta'), & \text{otherwise} \end{cases} \tag{2.15}$$

2.5.4.2 Policy Gradient Methods

The Policy Gradient Methods learn a parameterized policy and makes it possible to select the action without calculating the value function. Although, They may still compute action (or state) values, but they do not use them directly to select actions. Instead, the policy is represented directly, with its own weights independent of any value function.

The policy gradient theorem provides a formula for computing the gradient of the performance measure η with respect to the policy weights vector θ, composed of the state distribution $d_\pi(s)$, the state-action value function $q_\pi(s, a)$, and the gradient of the policy $\nabla_\theta \pi(a|s, \theta)$:

$$\nabla \eta(\theta) = \sum_a d_\pi(s) \sum_a q_\pi(s, a) \nabla_\theta \pi(a|s, \theta) \tag{2.16}$$

Actor-Critic methods

The policy gradient theorem provides an actor-critic architecture able to learn parameterized policies. The methods that exploit the advantage of both policy and value approximations are named actor-critic methods, where the *actor* estimates the policy weights vector for choosing an action and the *critic* estimates the value weights for providing the information about the quality of a state the agent ends up in after making the action according to the policy, as suggested by Figure 2.7.

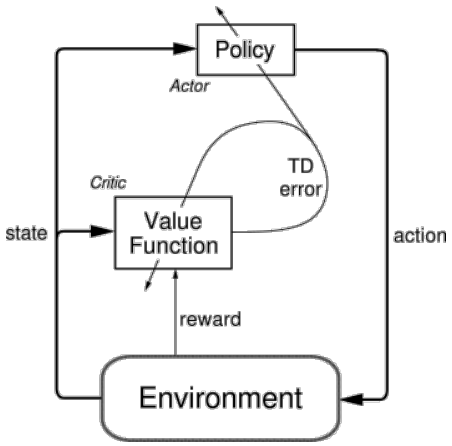


Figure 2.7: The Actor-Critic architecture [48]

Specifically, the critic updates the value function parameters, denoted w, and, depending on the algorithm, it could be action-value $\mathcal{Q}_w(\mathbf{s}, a)$ or state-value $V_w(\mathbf{s})$. In the meantime, the actor updates the neural network parameters θ for the policy $\pi_\theta(\mathbf{s}, a)$ in the direction suggested by the critic.

The actor-critic operation is detailed in Algorithm 4, where $\nabla_\theta \ln(\pi_\theta(\mathbf{s}, a))$ is the policy gradient, δ_t is the temporal difference (TD) error correction, and $\nabla_w \mathcal{Q}_w(\mathbf{s}, a)$ is the update of the Q-function parameters.

Algorithm 4 Actor-Critic

Input: The current state $\mathbf{s}^t$
Output: The best action a^t
Initialize $\mathbf{s}_t, \theta, w$; sample $a^t \sim \pi_\theta(\mathbf{s}, a)$
for $e = 1 \dots E_p$ **do**
  for $t = 1 \dots N$ **do**
    Sample reward $r_t \sim \mathcal{R}^e_{a^t}(\mathbf{s}^t, \mathbf{s}')$ and next state $\mathbf{s}' \sim P_{a^t}(\mathbf{s}^t, \mathbf{s}')$
    Sample the next action $a' \sim \pi_\theta(\mathbf{s}', a')$
    Update parameters $\theta \leftarrow \theta + \alpha_\theta \mathcal{Q}_w(\mathbf{s}, a) \nabla_\theta \ln(\pi_\theta(\mathbf{s}, a))$
    Compute $\delta_t = r_t + \gamma \mathcal{Q}_w(\mathbf{s}', a') - \mathcal{Q}_w(\mathbf{s}, a)$
    Update $w \leftarrow w + \alpha_w \delta_t \nabla_w \mathcal{Q}_w(\mathbf{s}, a)$
    Update $a \leftarrow a'$ and $\mathbf{s} \leftarrow \mathbf{s}'$
  end for
end for

2.5.5 Reinforcement Learning workflow

The general workflow for training an agent using reinforcement learning includes the following steps as described in figure 2.8

Figure 2.8: Reinforcement Learning workflow

1. **Formulate Problem** — Define the task for the agent to learn, including how the agent interacts with the environment and any primary and secondary goals the agent must achieve.

2. **Create Environment** — Define the environment within which the agent operates, including the interface between agent and environment and the environment dynamic model.

3. **Define Reward** — Specify the reward signal that the agent uses to measure its performance against the task goals and how this signal is calculated from the environment.
4. **Create Agent** — Create the agent, which includes defining a policy representation and configuring the agent learning algorithm.
5. **Train Agent** — Train the agent policy representation using the defined environment, reward, and agent learning algorithm.
6. **Validate Agent** — Evaluate the performance of the trained agent by simulating the agent and environment together.
7. **Deploy Policy** — Deploy the trained policy representation

Training an agent using reinforcement learning is an iterative process. Decisions and results in later stages can require you to return to an earlier stage in the learning workflow. For instance, if the training process does not converge to an optimal policy within a reasonable amount of time, you might have to update any of the hyper-parameters before retraining the agent.

Conclusion

The chapter discussed some background required for the work described in this report. First, we introduced the main characteristics of IoT networks as well as UAVs. Also, we provide some clustering techniques and in particular, the partitioning and hierarchical methods. Further, we presented Vehicle Routing Problem and different aspects to solve it. We then described the basics of Reinforcement Learning by explaining many important concepts associated with it.

In the next chapter, the problem of data collection using UAVs is addressed as we introduce the mathematical models. Then, the problem is formulated to be finally solved using different methods.

CHAPTER 3

PROPOSED MULTI-UAV DATA COLLECTION FOR IOT NETWORKS

Introduction

In this chapter, we propose a practical data collection approach for large-scale IoT systems, where multiple UAVs collect sensed data with different time deadlines by online and offline approaches. The rest of the chapter is structured as follows: First, we lay out a description of system model and the main assumptions. Next, we formulate the optimization problem covered by this research. Finally, we present our two-step approach and motivate the choices made.

3.1 System model

In this section, we present the considered network and channel models, the UAV-enabled data collection model, and the UAV energy model. All symbols used within this chapter are summarized in List of Symbols

3.1.1 Network and channel models

We consider an IoT sensor network consisting of a set $\mathcal{M} = \{1, \ldots, M\}$ of SNs, randomly located in a large area and constantly collecting time-sensitive data. We assume that SNs are heterogeneous, i.e., sensed data by SNs are of different

kind, and have different importance in time. Indeed, some sensed data are more critical and need to be collected faster than other less-critical ones.

Due to the limited energy and communication capability of typical IoT sensors, we assume that a set of $\mathcal{K} = \{1, \ldots, K\}$ cluster heads (CHs) can be deployed to collect, aggregate, and transmit sensed data to the collector. As shown in Fig. 3.1, collection of data is operated by multiple UAVs, which fly and hover above or close to CHs for a sufficient time to collect the aggregated data.

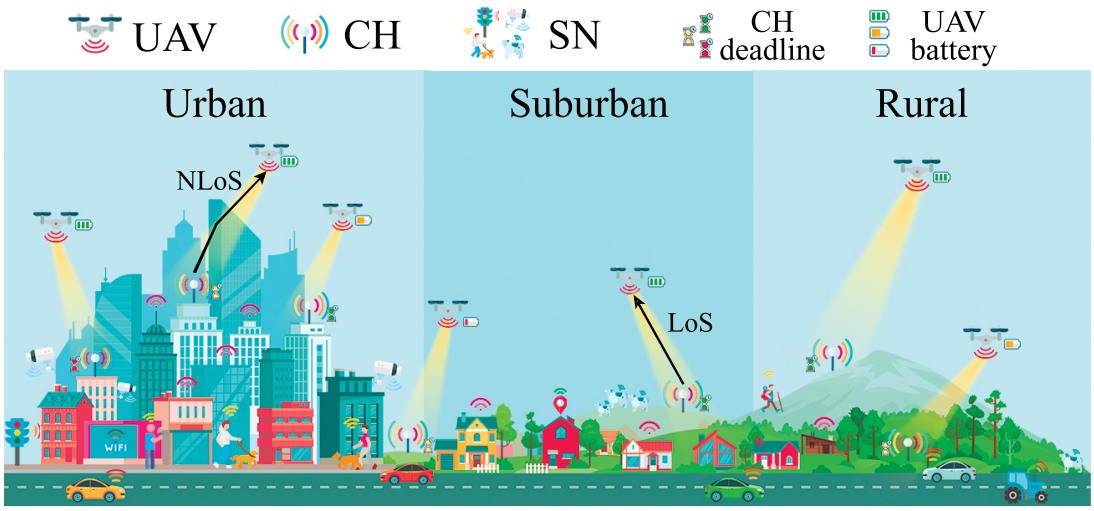


Figure 3.1: UAV-based data collection in clustered IoT networks with time deadlines

Each sensor transmits its sensed data to its associated CH using power P_{SN}, while a CH communicates with the collecting UAV using power $P_{\text{CH}} > P_{\text{SN}}$. Assuming that IoT sensors communicate with their associated CH using orthogonal channels, the received signal-to-noise ratio (SNR) at CH c for the signal transmitted by SN s, denoted γ_{sc}, can be written as

$$\gamma_{sc} = \frac{P_{\text{SN}} d_{sc}^{-\alpha}}{\sigma^2}, \forall s \in \mathcal{M},\ \forall c \in \mathcal{K} \tag{3.1}$$

where $d_{sc} = ||\mathbf{q}_s - \mathbf{q}_c||$ is the distance between IoT sensor s and CH c, $\mathbf{q}_i = [x_i, y_i, z_i]$ is the 3D location of node i ($i = s$ or c), α is the path loss exponent, and σ^2 is the noise power.

This communication link is considered successful if $\gamma_{sc} \geq \gamma_{\text{th}}$, where γ_{th} is a selected SNR threshold. Hence, a maximum communication range for each SN s associated with CH c is defined as [4]

$$d_{sc} \leq d_{\text{th}} = \left(\frac{P_{\text{SN}}}{\sigma^2 \gamma_{\text{th}}}\right)^{1/\alpha}, \forall s \in \mathcal{A}_c,\ c \in \mathcal{K} \tag{3.2}$$

where $\mathcal{A}_c$ is the set of SNs associated with CH c.

Given (3.2), the CHs are deployed to collect data from SNs and send it to UAVs. Let $\mathcal{U} = \{1, \ldots, U\}$ be the set of available UAVs. For the sake of simplicity, we assume that a UAV flies at a fixed altitude H such that this altitude respects authority regulations and safety considerations. Also, UAVs are equipped with sensors that allow obstacle avoidance at a certain safety distance, denoted d_{safe}, when they fly at maximum speed $v_{\max}$.

When UAV u hovers to receive data from CH c, the air-to-ground channel between them can be expressed using the probabilistic path loss model given by [49]

$$\Lambda_{cu} = Pr_{cu}^{\text{LoS}} l_{cu}^{\text{LoS}} + Pr_{cu}^{\text{NLoS}} l_{cu}^{\text{NLoS}}, \ \forall c \in \mathcal{K}_u, \ \forall u \in \mathcal{U}, \tag{3.3}$$

where Λ_{cu} is the average path-loss between CH c and UAV u, $\mathcal{K}_u \subset \mathcal{K}$ is the subset of ordered CHs to visit by UAV u, Pr_{cu}^{LoS} is the LoS probability, $Pr_{cu}^{\text{NLoS}} = 1 - Pr_{cu}^{\text{LoS}}$ is the NLoS probability, and l_{cu}^{LoS} and l_{cu}^{NLoS} are the LoS and NLoS path-losses, respectively. These parameters are written as [49]

$$Pr_{cu}^{\text{LoS}} = 1/\left(1 + ae^{b\ (\theta_{cu} - a)}\right), \ \forall c \in \mathcal{K}_u, \ \forall u \in \mathcal{U}, \tag{3.4}$$

and

$$l_{cu}^{m} = 20\log(4\pi f_c/V) + 20\log(d_{iu}) + \beta_m, \ m = \text{LoS or NLoS} \tag{3.5}$$

where a and b are constant values determined from the environment (rural, urban, etc.), $\theta_{cu} = \frac{180}{\pi} \times \arcsin\left(\frac{H}{d_{cu}}\right)$ is the elevation angle between UAV u and CH c, f_c is the carrier frequency, V is the speed of light, and β_m is the excessive path-loss coefficient. Using Friis formula, the received power at UAV u from CH c, denoted P_{cu}, is expressed by

$$P_{cu} = P_{\text{CH}} - \Lambda_{cu}, \ \forall c \in \mathcal{K}_u, \ \forall u \in \mathcal{U}, \tag{3.6}$$

where P_{CH} is set, for the sake of simplicity, to 0 dBm.

3.1.2 UAV data collection model

We assume that the mission time, i.e., the maximum time for UAVs to collect data from CHs and return to their dockstation, is T_F and divided into N time slots (TSs). Thus, TSs are equal and are of length $\delta = \frac{T_F}{N}$. We define the 3D location of UAV u in time slot t as $\mathbf{q}_u^t = [x_u^t, y_u^t, H]$. Since the UAV's speed is

limited by $v_{\max}$, its traveled distance in one TS is constrained by

$$||\mathbf{q}_u^{t+1} - \mathbf{q}_u^t|| = \sqrt{(x_u^{t+1} - x_u^t)^2 + (y_u^{t+1} - y_u^t)^2} \leq v_{\max}\delta. \tag{3.7}$$

In our system, each CH c uploads its data to its associated UAV u at rate R_{cu} in bits/second (bps), determined using the Shannon equation as follows:

$$R_{cu} = W \log_2(1 + \gamma_{cu}),\ \forall c \in \mathcal{K}_u,\ \forall u \in \mathcal{U}, \tag{3.8}$$

where W is the total bandwidth of the channel and $\gamma_{cu} = \frac{\bar{P}_{cu}}{\sigma^2}$ is the SNR, where $\bar{P}_{cu} = 10^{\frac{P_{cu}}{10}}$ is the linear value of P_{cu}. Accordingly, the required number of TSs to upload one packet to the UAV can be expressed by

$$T_{1,cu} = \left\lceil \frac{S_p}{R_{cu}\delta} \right\rceil,\ \forall c \in \mathcal{K}_u,\ \forall u \in \mathcal{U}, \tag{3.9}$$

where S_p is the size of the packet in bits and $\lceil \cdot \rceil$ is the ceiling function. Given Q_c data packets to be transmitted by CH c, then the required data collection time (in TSs) by UAV u is

$$T_{cu} = Q_c T_{1,cu},\ \forall c \in \mathcal{K}_u,\ \forall u \in \mathcal{U}. \tag{3.10}$$

Due to the different priorities of collected data at each CH, we define by $T_{d,c}$ the time deadline of data stored in CH c, i.e., if not totally collected within this deadline, the data present in the CH becomes outdated and is dropped[1]. Hence, data collection has to respect the following constraint, assuming that all packets in one CH are collected by a single UAV:

$$\sum_{m=1}^{\phi_u(c)} [T_{mu} + T_{f,u}(m-1, m)] \leq T_{d,c},\ \forall c \in \mathcal{K}_u, \tag{3.11}$$

where $\phi_u(c)$ designates the rank (i.e., index) of CH c in $\mathcal{K}_u$, T_{mu} is the data collection time of the m^{th} ranked CH in $\mathcal{K}_u$, and $T_{f,u}(m-1, m)$ is the flight time (in TSs) from the hovering location to collect data from the $(m-1)^{th}$ ranked CH to that associated to the m^{th} ranked CH in $\mathcal{K}_u$, with $m = 0$ being the dockstation. The flight time of UAV u between two locations $\mathbf{q}_{u,i}$ and $\mathbf{q}_{u,i'}$, where i and i' are two CHs ranks in the UAV path, can be calculated as follows:

$$T_{f,u}(i, i') = \frac{||\mathbf{q}_{u,i'} - \mathbf{q}_{u,i}||}{v_u\ \delta}, \tag{3.12}$$

[1]Typically, this deadline is determined by the most critical sensed data. It sets our system to stringent conditions, where all data of a CH are considered critical.

where $v_u \leq v_{\max}$ is the flying speed of the UAV (in m/sec).

3.1.3 UAV energy model

The UAV is energy-constrained due to limited on-board battery. The battery lifetime depends on several factors, e.g., UAV's energy source, type, weight, speed, etc. Typically, the UAV's energy consumption consists of the propulsion energy and the communication energy. Without loss of generality, communication energy is several orders of magnitude smaller than propulsion energy. Hence, it is neglected in the considered energy model. For the propulsion energy, we adopt the propulsion-power model for rotary-wing UAVs as follows [50]:

$$P_{\text{prop,u}}(v_u) = \zeta_I \left(\sqrt{1 + \frac{v_u^4}{4v_0^4}} - \frac{v_u^2}{2v_0^2} \right)^{1/2} + \zeta_B \left(1 + \frac{3v_u^2}{U_{\text{tip}}^2} \right) + \frac{1}{2} d_0 \psi r A v_u^3, \quad (3.13)$$

where ζ_B and ζ_I are the blade profile power and induced power, respectively. U_{tip} is the tip speed of the rotor blade, v_0 is the mean rotor induced velocity, ψ is the air density, d_0 is the fuselage drag ratio, r is the rotor solidity, and A denotes the rotor disc area. Consumed power at hovering is obtained for speed $v_u = 0$, and written as

$$P_{\text{hov,u}} = P_{\text{prop,u}}(v_u = 0) = \zeta_B + \zeta_I. \quad (3.14)$$

Using (3.13)–(3.14), the consumed energy of UAV u during a time period T can be given by

$$E_u(v, T) = \begin{cases} P_{\text{prop,u}}(v) \times T, & \text{if } v > 0 \\ P_{\text{hov,u}} \times T, & \text{if } v = 0. \end{cases} \quad (3.15)$$

Hence, the related battery status at TS t, denoted $S_u(t)$, can be expressed by

$$S_u(t) = S_u(t-1) - E_u(v, \delta), \ \forall t > 1, \quad (3.16)$$

where $S_u(t-1)$ is the battery's status at the end of TS $(t-1)$, and $S_u(0)$ is the initial battery capacity. The latter is expressed as $S_u(0) = S_{\text{ini}} + S_{\min}$, where S_{ini} is the battery capacity dedicated for the mission, while $S_{\min}$ is a safety capacity, reserved for emergency pull back to the dockstation. Consequently, we have $S_u(t) \in [S_{\min}, S_u(0)]$.

3.2 Problem formulation

In this section, we formulate our optimization problem aiming to maximize an utility function of data collection, penalized by the UAVs' energy consumption.

Let $K_u = |\mathcal{K}_u|$ be the cardinality of the set $\mathcal{K}_u$, $\forall u \in \mathcal{U}$. Then, using (3.15), the consumed energy by UAV u during the mission is given by

$$E_u(\mathcal{K}_u) = \sum_{m=1}^{K_u+1} \left[E_u(0, T_{mu}\,\delta) + E_u(v_u, T_{f,u}(m-1,m)\,\delta)\right], \tag{3.17}$$

where $T_{f,u}(K_u, K_u+1) = \|\mathbf{q}_{u,K_u} - \mathbf{q}^0\|/v_u\delta$ and $\mathbf{q}^0$ is the initial location of all UAVs, i.e., the dockstation. Therefore, the optimization problem can be formulated as follows:

$$\max_{\substack{\mathcal{K},\mathcal{L},\{\mathcal{A}_c\}_{c\in\mathcal{K}} \\ \mathcal{U},\{\mathcal{K}_u,\mathcal{L}_u\}_{u\in\mathcal{U}}}} \quad \sum_{u=1}^{U} \left[f_1(\mathcal{K}_u) - f_2(E_u(\mathcal{K}_u))\right] \tag{P1}$$

$$\text{s.t.} \quad d_{sc} \le d_{\text{th}},\ \forall s \in \mathcal{A}_c, \forall c \in \mathcal{K}, \tag{P1.a}$$

$$|\mathcal{A}_c| \le F,\ \forall c \in \mathcal{K}, \tag{P1.b}$$

$$S_u(t) \ge S_{\min},\ \forall u \in \mathcal{U}, \forall 1 \le t \le T_u,\ \forall u \in \mathcal{U} \tag{P1.c}$$

$$\sum_{m=0}^{\phi_u(c)} (T_{mu} + T_{f,u}(m-1,m)) \le T_{d,c}, \tag{P1.d}$$

$$\|\mathbf{q}_u^t - \mathbf{q}_{u'}^t\| \ge d_{\text{safe}}, \forall t \ge 1, \forall (u,u') \in \mathcal{U}^2,\ u \ne u' \tag{P1.e}$$

$$\|\mathbf{q}_u^{t+1} - \mathbf{q}_u^t\| \le v_{max}\delta,\ \forall 1 \le t \le T_u,\ \forall u \in \mathcal{U}, \tag{P1.f}$$

where f_1 and f_2 are monotonically increasing functions that emphasize the reward and penalty for collecting data from the CHs and consuming energy to reach them, respectively. Also, $|\mathcal{A}_c|$ is the cardinality of the set $\mathcal{A}_c$, $T_u \le N$ is the effective mission completion time for UAV u (in TSs), $\mathcal{L} = \{\mathbf{q}_c\}_{c\in\mathcal{K}}$ is the set of selected locations for CHs, and $\mathcal{L}_u = \{\mathbf{q}_{u,i}\}_{i\in\mathcal{K}_u}$ is the set of ordered UAV hovering locations to collect data from the associated CHs.

The constraint (P1.a) ensures the successful communication between SNs in $\mathcal{A}_c$ and their associated CH c, whereas (P1.b) guarantees fairness in associating SNs with CHs, where F is the maximum number of SNs associated with one CH. (P1.c) makes sure that enough energy is available in the battery to complete the mission at any TS t, while (P1.d) satisfies the time deadline condition when a UAV collects data from CHs. Also, (P1.e) ensures that no collisions between UAVs occur, and finally, (P1.f) limits the flying distance for a given $v_{\max}$.

The formulated problem is NP-hard. Indeed, in the special case of already deployed CHs, (P1) is equivalent to determining the best paths for UAVs, while

respecting the battery and time deadlines conditions.

In (P1), we see that the optimization of parameters $\mathcal{U}$ and $\{\mathcal{K}_u, \mathcal{L}_u\}_{u\in\mathcal{U}}$ depend on the selected parameters $\mathcal{K}, \{\mathcal{A}_c\}_{c\in\mathcal{K}}$ and $\mathcal{L}$. Also, since their constraints are independent, (P1) can be divided into two cascaded problems as follows:

1. A first problem of IoT sensors clustering and CHs placement, where only $\mathcal{K}$, $\mathcal{L}$, and $\{\mathcal{A}_c\}_{c\in\mathcal{K}}$ are optimized, aiming to minimize the number of deployed CHs is formulated. As we notice that this objective has a direct impact on the consumed energy of data collection since a lower number of CHs would reduce the mission time and energy consumption of UAVs.

2. Then, given $\mathcal{K}$, $\mathcal{L}$ and $\{\mathcal{A}_c\}_{c\in\mathcal{K}}$, a second problem of UAVs trajectory planning is formulated in order to to maximize data collection while consuming the minimum UAV energy, through the optimization of $\mathcal{U}$ and $\{\mathcal{K}_u, \mathcal{L}_u\}_{u\in\mathcal{U}}$.

Hence, we solve these two sub-problems in the next two sections.

3.3 Proposed IoT sensors clustering solution

Typically, IoT sensors are deployed in large areas in order to sense, process, and communicate relevant data for some applications. Due to their limited battery capacities, IoT sensors need to carefully use their energy, while prolonging the life of collected data as much as possible. To do so, IoT sensors can be grouped into disjoint and non-overlapping clusters to reduce the amount of used energy. SNs scan the surrounding environment and transmit sensed data to CHs, which aggregate and transmit obtained information to the UAVs [51].

3.3.1 Data aggregation

Data aggregation is the process of integrating multiple copies of from diverse sensors into one copy, since the information reported by the neighbouring nodes has some degree of redundancy and highly correlated. The introduction of data aggregation usually involves the fusion of data from multiple sensors at intermediate nodes and transmitting it to the base station or to UAVs [52].

Data aggregation is a widely used technique in IoT networks. Its main objective is to reduce the required communication at different levels and and maximize overall network lifetime. Thus, it is considered as one of the fundamental processing procedures to decrease the total energy consumption. Such approach is considered energy efficient since we can reduce the number of transmitted data packets, and as the energy consumed in direct transmission to the base station is much greater

as compared to energy consumption for aggregation. As a result, we benefit both from saving energy and obtaining accurate information, while sacrificing performance in other areas. Hence, there are further factors which determine the energy efficiency of a sensor network such as network architecture, the data aggregation method and the underlying clustering protocols.

3.3.2 IoT sensors clustering

Several state-of-the-art clustering techniques exist, e.g., K-means [53], density-based spatial clustering, and Hierarchical Agglomerative Clustering (HAC) [41]. Nevertheless, due to the low-complexity and high scalability of K-means, we focus here on its customization, aiming to group SNs and deploy the minimal number of CHs. Thus, the associated clustering problem can be written as

$$\min_{\substack{\mathcal{K},\mathcal{L},\\ \{\mathcal{A}_c\}_{c\in\mathcal{K}}}} \quad \sum_{c=1}^{K}\sum_{s=1}^{M} a_{sc}\, d_{sc}^2 \tag{P2}$$

$$\text{s.t.} \quad d_{sc} \leq d_{\text{th}}, \forall s \in \mathcal{A}_c, \forall c \in \mathcal{K} \tag{P2.a}$$

$$|\mathcal{A}_c| \leq F, \quad \forall c \in \mathcal{K}, \tag{P2.b}$$

where $a_{sc} \in \{0, 1\}$ is a binary variable indicating whether SN s is associated with CH c or not, $\forall s \in \mathcal{A}_c$.

Conventionally, K-means clustering requires a predefined number of CHs. Hence, in [4], the authors proposed a K-means algorithm to solve (P2) without any constraint, then reexecuted it iteratively until (P2.a) is met. Such method is time consuming and inaccurate, as the clustering performance may vary with the algorithm initialization. Therefore, we propose here to improve this method by integrating both constraints (P2.a)–(P2.b) into the clustering process. The proposed approach, presented in Algorithm 5, is described as follows. First, locations of K CHs are randomly initialized. Then, each SN is assigned to the closest CH with respect to (P2.a)–(P2.b). Next, the locations of CHs are updated by calculating the resulting mean location of associated SNs for each CH. This procedure is repeated until convergence, i.e., the calculated locations remain unchanged. Since we aim to deploy the minimal number of CHs, we execute the aforementioned steps for an increasing number of CHs until a solution to (P2) is obtained, i.e., K is determined. Although the calculated CHs locations are an adequate solution for (P2), this may not be the case for the main problem (P1).

In this context, we propose to improve the locations of CHs by making them closer to the dockstation. Indeed, CH c has a mobility margin if its furthest

associated SN, denoted s_0, respects (P2.a) loosely, i.e., $d_{s_0c} < d_{\text{th}}$.

Hence, making the CHs closer to the dockstation would increase the probability to respect data collection deadlines, shorten the mission time, and improve the energy consumption of UAVs. This procedure is implemented in lines 26-27 of Algorithm 5.

Algorithm 5 IoT sensors clustering algorithm

1: **Input:** $\mathcal{M}$, $\{\mathbf{q}_s\}_{s\in\mathcal{M}}$, and $\max_{\text{it}}$ %$\max_{\text{it}}$ is the max. number of CHs
2: **Output:** Optimal $\mathcal{K}$, $\mathcal{L}$, and $\{\mathcal{A}_c\}_{c\in\mathcal{K}}$
3: Set $k = (M \div F)$ % F is the max. number of SNs in a cluster
4: **for** k **to** $\max_{\text{it}}$ **do**
5: Set $\mathcal{L}_{old} = \{\}$, set $\mathcal{L}$ randomly, and set $\mathcal{A}_c = \emptyset$, $\forall c = 1, \ldots, k$
6: Set $error = dist(\mathcal{L}, \mathcal{L}_{old})$ %Convergence parameter
7: **while** $error \neq 0$ **do**
8: **for** $s \in \mathcal{M}$ **do**
9: Calculate d_{sc}, $\forall c = 1, \ldots, k$
10: Find $c_0 = \arg(\min(d_{sc}))$
11: **if** $d_{sc_0} \leq d_{\text{th}}$ & $|\mathcal{A}_{c_0} \cup \{s\}| \leq F$ **then**
12: $\mathcal{A}_{c_0} = \mathcal{A}_{c_0} \cup \{s\}$
13: **else**
14: $\mathcal{A}_{k+1} = \mathcal{A}_{k+1} \cup \{s\}$
15: **end if**
16: **end for**
17: Set $\mathcal{L}_{old} = \mathcal{L}$
18: Calculate $\mathcal{L}$ as the mean location of the associated SNs in $\mathcal{A}_c$
19: Calculate $error = dist(\mathcal{L}, \mathcal{L}_{old})$
20: **end while**
21: **if** $\mathcal{A}_{k+1} = \emptyset$ **then**
22: $\mathcal{K} = \{1, \ldots, k\}$
23: Break
24: **end if**
25: **end for**
26: **while** $\max(d_{sc}) < d_{\text{th}}$, $\forall s \in \mathcal{A}_c$ **do**
27: Get $\mathcal{L}$ closer to the dockstation
28: **end while**

3.4 Proposed Multi-UAV path planning solutions

3.4.1 Offline path planning: Meta-heuristic approaches

Problem (P1) is NP-hard. Indeed, in the special case of already deployed CHs, the problem becomes finding the best routes for UAVs, while respecting the energy and time deadlines conditions. The latter can be seen as the capacitated vehicle

routing problem with time windows (CVRPTW) [9]. The CVRPTW can be described as selecting the routes for a number of vehicles, aiming to serve a group of customers within time windows. Each vehicle has a limited capacity, which is used to depart from a depot point, serve a number of customers along its route, then return to the same depot point. The objective of the CVRPTW is to minimize the total transport costs. Logically, the vehicles, customers, and transport costs, can be assimilated by the UAVs, CHs, and energy consumption, respectively.

Given $\mathcal{K}$, $\mathcal{L}$, and $\mathcal{A}_c$, $\forall c \in \mathcal{K}$, the trajectory planning problem can be formulated as

$$\max_{\substack{\mathcal{U}, \\ \{\mathcal{L}_u, \mathcal{K}_u\}_{u\in\mathcal{U}}}} \quad \sum_{u=1}^{U} [f_1(\mathcal{K}_u) - f_2(E_u(\mathcal{K}_u))] \tag{P3}$$

$$\text{s.t.} \quad S_u(t) \geq S_{\min},\ \forall u \in \mathcal{U}, \forall 1 \leq t \leq T_u,\ \forall u \in \mathcal{U} \tag{P3.a}$$

$$\sum_{m=0}^{\phi_u(c)} (T_{mu} + T_{f,u}(m-1, m)) \leq T_{d,c}, \tag{P3.b}$$

$$\|\mathbf{q}_u^t - \mathbf{q}_{u'}^t\| \geq d_{\text{safe}}, \forall t \geq 1, \forall (u, u') \in \mathcal{U}^2,\ u \neq u' \tag{P3.c}$$

$$\|\mathbf{q}_u^{t+1} - \mathbf{q}_u^t\| \leq v_{max}\delta,\ \forall 1 \leq t \leq T_u,\ \forall u \in \mathcal{U}. \tag{P3.d}$$

In order to solve it, we model our system as a graph, described as follows.

Let $G = (\mathcal{D}, \mathcal{E})$ be a complete graph, where $\mathcal{D} = \{0, 1, \ldots, K\}$ is a set of vertices (nodes) representing the dockstation (node 0) and K CHs, and $\mathcal{E}$ is the set of directed edges connecting the nodes. A directed edge from node i to node j, denoted e_{ij}, represents the UAV's flying operation from i to j and the hovering operation at j.

A cost associated to the edge e_{ij}, called $\chi_u(e_{ij}) = E_u(v_u, \mu_1(e_{ij})) + E_u(0, \mu_2(e_{ij}))$, is expressed as the sum of the UAV's flying and hovering energy, where $\mu_1(e_{ij}) = T_{f,u}(i, j)\delta$ and $\mu_2(e_{ij}) = T_{ju}\delta$ respectively, $\forall u \in \mathcal{U}$.[2]

Moreover, the time deadlines at CHs are represented in the graph by a time window $\omega_c = [o_c, v_c]$, where o_c and v_c are the minimum and maximum instants for data collection, for each node $c \in \mathcal{D}$. This time window defines when data collection at node c can begin and end. Finally, trajectory of UAV u in $\mathcal{G}$ can be defined by $\mathcal{K}_u$, where each element of $\mathcal{K}_u$ is an ordered node to visit. Accordingly,

[2]This is valid assuming that all UAVs are of the same type, i.e., having the same mechanical and communication characteristics.

(P3) can be reformulated as problem (P4) detailed below:

$$\min_{\substack{\mathcal{U},\{\mathcal{K}_u,\\ \mathcal{L}_u\}_{u\in\mathcal{U}}}} \quad \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{D}}\sum_{j\in\mathcal{D}} \chi_u(e_{ij})\; b_{u,ij} \tag{P4}$$

$$\text{s.t.} \quad \sum_{u=1}^{U}\sum_{i\in\mathcal{D}} b_{u,ij} \leq 1, \quad \forall j \in \mathcal{D} \tag{P4.a}$$

$$\sum_{j\in\mathcal{D}} b_{u,0j} = \sum_{i\in\mathcal{D}} b_{u,i0} = 1, \quad \forall u \in \mathcal{U} \tag{P4.b}$$

$$\sum_{i=1}^{j-1} \mu_1(e_{i(i+1)}) + \mu_2(e_{i(i+1)}) \in [o_c, v_c],\ \forall j \in \mathcal{K}_u \tag{P4.c}$$

$$\sum_{i\in\mathcal{D}}\sum_{j\in\mathcal{D}} \chi_u(e_{ij}) b_{u,ij} \leq S_u(0) - S_{\min},\ \forall u \in \mathcal{U}, \tag{P4.d}$$

where $b_{u,ij}$ is the binary variable indicating the selection of the edge $i-j$ within UAV u's trajectory. Constraint (P4.a) ensures that each CH is visited at most once by exactly one UAV. (P4.b) guarantees that each UAV departs and returns to the dockstation. Condition (P4.c) emphasizes that the data collection time cannot exceed the time deadline. Finally, (P4.d) guarantees that the consumed energy by any UAV respects the battery capacity. Subsequently, problem (P4) can be solved using any of the available CVRPTW heuristic or meta-heuristic approaches, such as gradient descent, simulated annealing, Tabu search, etc. [54].

3.4.2 Online path planning: Reinforcement-Learning based approaches

In contrast to offline approaches, online algorithms are commonly used when inputs or data are arriving in a sequential order and without being fully available. In our context, DRL-based algorithms can be used for path planning. To do so, we start by reformulating the optimization problem (P3) as a Markov Decision Process (MDP).

An MDP is a discrete-time stochastic control process used for a variety of optimization problems, where the outcome is partially random and under the control of a decision maker [55]. The state of the process in slot t is denoted $\mathbf{s}^t$ and the decision maker may choose any action a^t in $\mathcal{A}(\mathbf{s}^t)$, which is the set of available actions in state s^t. Subsequently, the system transits to a new state $\mathbf{s}'$ and provides the reward $\mathcal{R}_{a^t}(\mathbf{s}^t, \mathbf{s}')$. State $\mathbf{s}'$ depends on the current state $\mathbf{s}^t$ and the taken action. The transition probability to state $\mathbf{s}'$ is given by the function $P_{\mathbf{a}^t}(\mathbf{s}^t, \mathbf{s}')$. Specifically, the MDP's state transition in our system is a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$

defined as follows:

- $\mathcal{S} = \{\mathbf{s}^t\}$ is a set of states $\mathbf{s}^t = \{\mathbf{q}_u^t, \mathcal{H}_u^t\}$. The latter is composed of the UAV u's location $\mathbf{q}_u^t$ and the number of visited CHs $\mathcal{H}_u^t$ by the u^{th} UAV.
- $\mathcal{A} = \{a^t\}$ is the set of actions, where $a^t \in \{0 \dots 7\}$ represents the set of cardinal and inter-cardinal directions, i.e., 0 (North), 1 (South), 2 (East), 3 (West), 4 (Northeast), 5 (Northwest), 6 (Southeast), and 7 (Southwest).

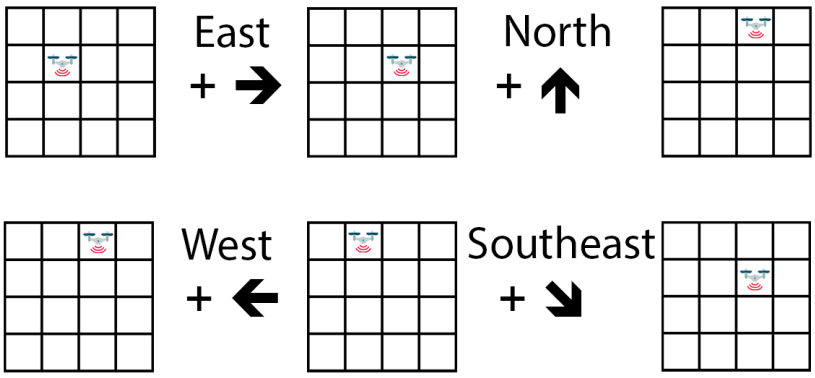


Figure 3.2: Example of actions in the gridworld simulations

- $\mathcal{P} = \{P_{a^t}(\mathbf{s}^t, \mathbf{s}')\}$ is the set of transition probabilities.
- $\mathcal{R}$ is the immediate reward value, obtained when action a^t is taken at state $\mathbf{s}^t$. Its expression is related to the objective function and constraints defined in (P3). Specifically, the reward function is designed to maximize collected data within the deadlines while minimizing the energy consumption, with respect to constraints (P1.c) – (P1.f). Hence, the reward function is defined as

$$\mathcal{R}_{a_t}^e(\mathbf{s}_t, \mathbf{s}') = f_{1,t}^e(\mathcal{K}_u) - f_{2,t}^e(E_u(\mathcal{K}_u)) \tag{3.24}$$

where $f_{1,t}^e$ and $f_{2,t}^e$ are two functions that describe the reward and penalty in episode e for collecting data from the CHs and consuming energy to reach them, respectively. They are given by

$$f_{1,t}^e(\mathcal{K}_u) = \begin{cases} \Phi_1 & \text{if } t \leq T_{d,c},\ \forall c \in \mathcal{K}_u \\ \Phi_2 & \text{if } t > T_{d,c},\ \forall c \in \mathcal{K}_u, \end{cases} \tag{3.25}$$

and

$$f_{2,t}^e(E_u(\mathcal{K}_u)) = \begin{cases} \Psi_1 & \text{if } \mathbf{q}_u^t \notin \mathcal{S} \\ \Psi_1 + \Psi_2 & \text{if } \mathbf{q}_u^t \in \mathcal{S}, \end{cases} \tag{3.26}$$

where $f_{1,t}^e(\mathcal{K}_u)$ offers a reward Φ_1 if UAV u collects data from CH c be-

fore expiration of the deadline $T_{d,c}$, otherwise it gives a penalty Φ_2, while $f^e_{2,t}(E_u(\mathcal{K}_u))$ minimizes the energy consumption by imposing a penalty Ψ_1 on the UAV's motion, aiming to reduce the trajectory length. An additional penalty Ψ_2 can be applied to prevent the UAV from revisiting the same location $\mathbf{q}^t_u$ during its flight.

- $\gamma \in [0, 1]$ is the discount factor.

Our objective is to train a policy $\pi^*(\mathbf{s})$ that maximizes the discounted cumulative reward $\mathcal{R}^e$ of each episode e is expressed by

$$\mathcal{R}^e = \sum_{t=1}^{N} \gamma^t \mathcal{R}^e_{a_t} = f^e_1(\mathcal{K}_u) - f^e_2(E_u(\mathcal{K}_u)), \tag{3.27}$$

where N denotes the total number of steps in one episode e, $f^e_1(\mathcal{K}_u) = \sum_{t=1}^{N} \gamma^t f^e_{1,t}(\mathcal{K}_u)$, and $f^e_2(E_u(\mathcal{K}_u) = \sum_{t=1}^{N} \gamma^t f^e_{2,t}(E_u(\mathcal{K}_u)$.

Hence, we propose novel DRL based approaches where a single multi-task agent that has no knowledge about the CHs' locations and their data deadlines, attempts to determine the sequential actions to collect the maximum amount of data using multiple UAVs. Specifically, we adopt DQN, DDQN, and AC based algorithms.

Conclusion

In this chapter, we discussed the main elements of the system model which should be optimized. Then, we presented in details our efficient approaches to solve them. In next chapter we discuss our evaluation and experimental results regarding the proposed approach.

CHAPTER 4

RESULTS AND DISCUSSION

Introduction

In this chapter, we evaluate the performance of the proposed solutions for IoT sensors clustering and UAV path planning. The proposed approaches are tested in several case scenarios to evaluate their strengths and drawbacks. Namely, the impact of time deadlines and battery capacity on the total consumed energy.

4.1 IoT sensors clustering results

Unless specified otherwise, we assume M=2000 SNs randomly and uniformly distributed within a geographical area of 5×5 km^2. The remaining simulation parameters are presented in Table 4.1.

Table 4.1: Simulation parameters

UAV altitude	H=100 m	Bandwidth	W=10 MHz	Speed of light	V=3.10^8 m/sec
UAV speed	v_u=30 m/sec	Speed of light	V=3.10^8 m/sec	Distance threshold	d_{th}=600 m
Environment	Urban	Mission time	T_f=300 sec	IoT devices distribution	Uniform
Carrier frequency	f_c=2 GHz	Environment parameters	a=9.61 b=0.16	Time deadline	$T_{d,c}$=140 sec
Max SNs per cluster	F=120	Noise power	σ^2=-109 dBm	Packet size	S_p=1 Kbyte
TS length	δ=100 msec	Path loss exponent	α=2.7	Number of packets	Q_c=50000
Blade profile power	ζ_B=3.4 W	Induced power	ζ_I=118 W	Tip speed of rotor blade	U_{tip}=60 m/sec
Mean rotor induced velocity	v_0=5.4 m/sec	Air density	ψ=1.225 Kg/m^3	Fuselage drag ratio	d_0=0.3
Rotor solidity	r=0.03	Rotor disc area	A=0.28 m^2	Battery capacity	$S_u(0)$=3500 mAh

Fig. 4.1 presents the clustering performance of several algorithms, expressed by the average number of clusters as a function of communication range, d_{th} (averaging over 100 scenarios of SNs). We consider three clustering algorithms, namely proposed Algo. 5, K-means-based clustering [4], and Hierarchical Agglomerative Clustering (HAC [40]).

As d_{th} increases, a smaller number of CHs is needed, since the latter can be reached by a higher number of SNs. Moreover, Algo. 5 achieves the best clustering performance. Indeed, unlike other algorithms, Algo. 5 is capable of adjusting the clusters to constraints (P2.a)–(P2.b) on-the-fly, i.e., while processing the SNs.

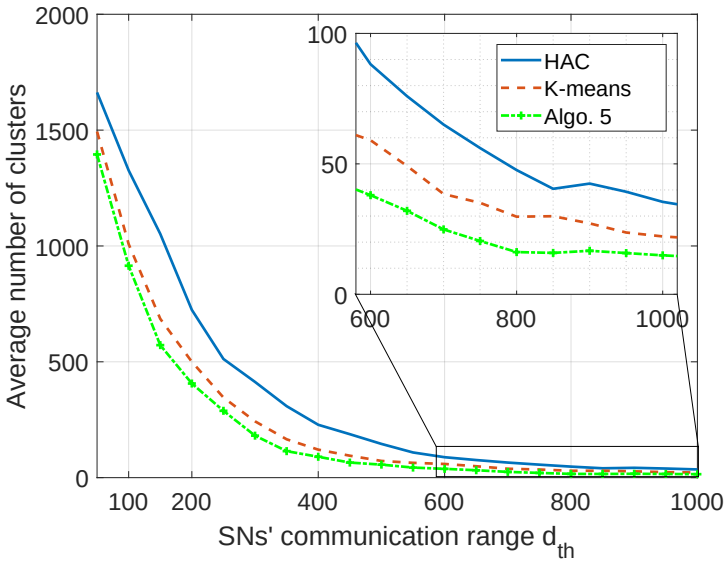


Figure 4.1: Average number of clusters vs. d_{th}

In Fig. 4.2, we depict a scenario where Algo. 5 is used to group the SNs into 48 clusters, from which data is collected using 12 UAVs.

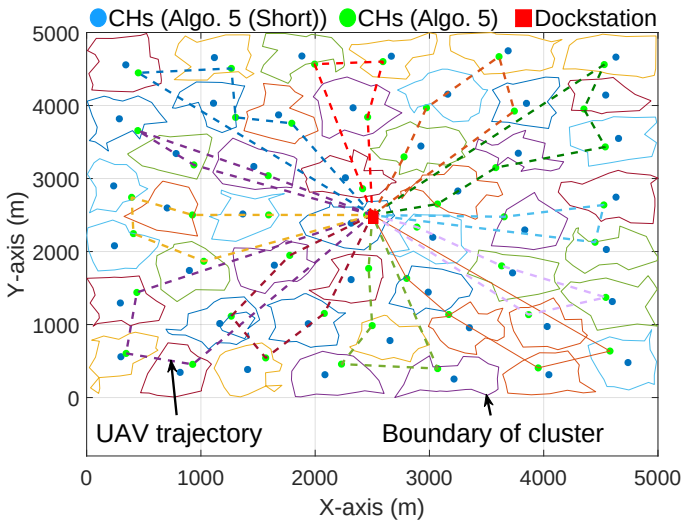


Figure 4.2: Clustering and UAV trajectories

We distinguish two variations, namely Algo. 5 as is, and "Algo. 5 (Short)", where the algorithm lines 26–27 are omitted. It is obvious that Algo. 5 would result in a more energy-efficient data collection, since UAVs would fly for less time, assuming that they hover directly above each CH to collect data.

4.2 Offline path planning results

For the SNs' and CHs' design of Fig. 4.2 (Algo. 5), Figs. 4.3–4.4 compare the consumed energy (in kilojoule -kJ), number of visited CHs, and number of deployed UAVs performances, as functions of the battery capacity, for different offline UAV trajectory planning approaches, namely Tabu search (Tabu), simulated annealing (SA), and guided local search (GLS).

Performances of these methods are almost similar, with a slight preference for Tabu. Indeed, the latter consumes less or equal energy to SA and GLS. Also, as the battery capacity increases (above 2250 mAh), the performance saturates, thus guaranteeing that data is collected from all CHs. Finally, the optimal battery capacity of 2500 mAh is provided by Tabu, where minimum energy (=167.9 kJ) and number of UAVs (=12) are achieved.

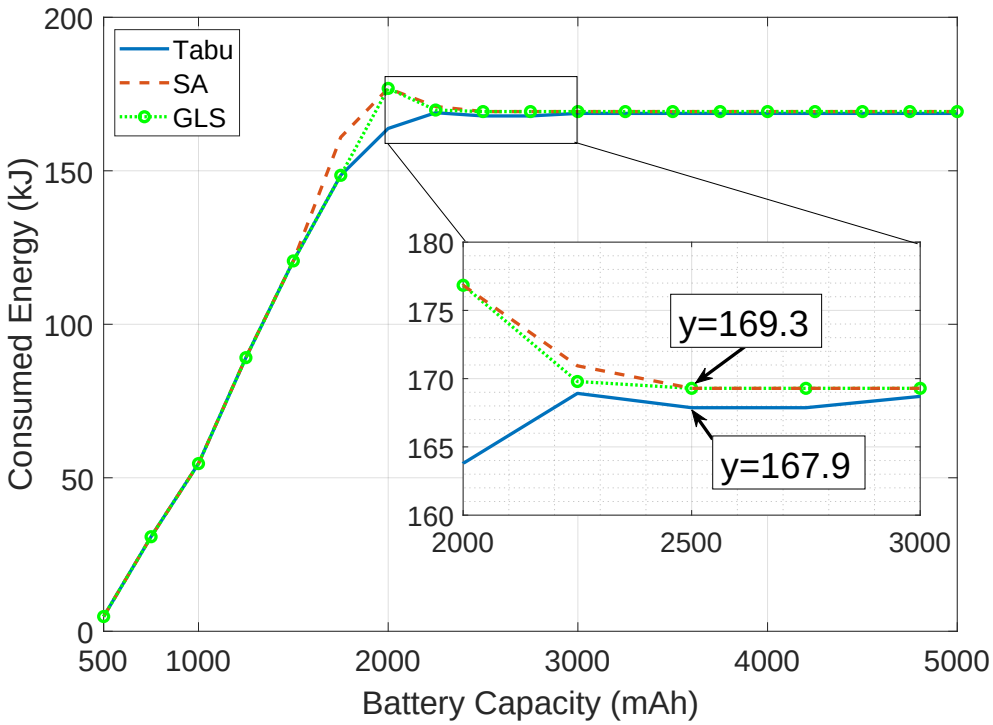


Figure 4.3: Energy consumption vs. battery capacity

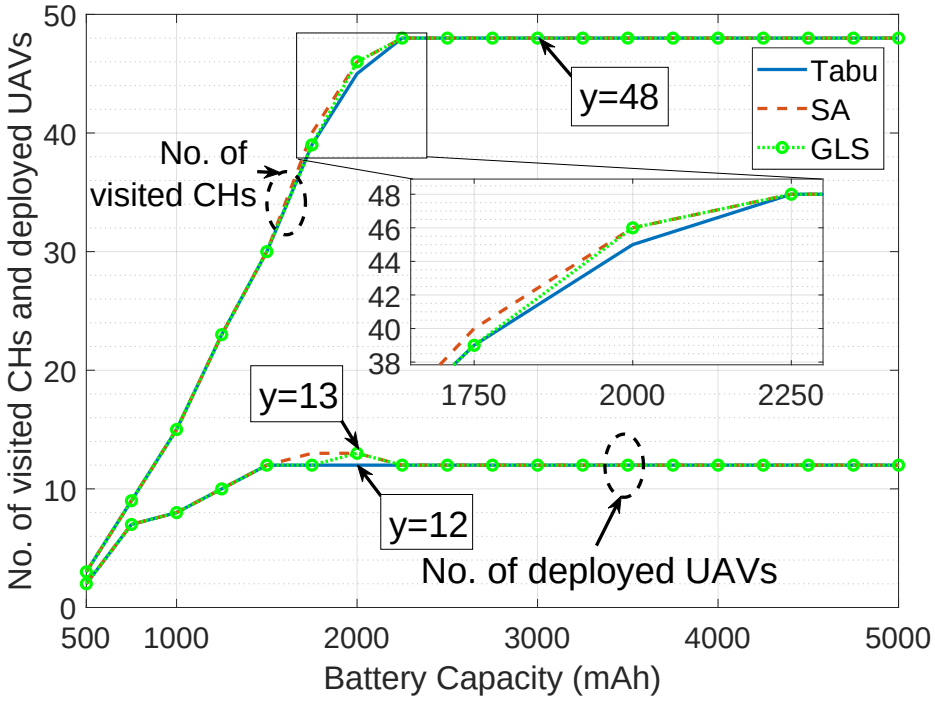


Figure 4.4: Number of CHs and UAVs vs. battery capacity

In Figs. 4.5–4.6, Tabu algorithm performances are depicted as functions of the time deadline, for combinations of IoT sensors clustering methods and UAV hovering techniques, namely, a combination of Algo. 5 and UAVs hovering directly above the CHs for data collection, denoted by "Algo. 5 + HA", a combination of "Algo. 5 (Short)" and HA, called "Algo. 5 (Short) + HA", and finally a combination of Algo. 5 and UAVs hovering within a coverage range of 168 m from the CHs while guaranteeing reliable communication, is designated as "Algo. 5 + HR".

For all approaches, as the deadline increases, the consumed energy reaches a peak then decreases. Below this peak, the deadline is very small such that it prevents collecting data from all CHs as the latter will be outdated, hence, a small number of UAVs is deployed as shown in fig. 4.6, which consumes low energy due to short trajectories. In contrast, beyond the peak point, the deadline is long enough to deploy few UAVs that visit all CHs. The peak point corresponds to the critical deadline for which the maximum number of UAVs is deployed to collect data from the CHs. "Algo. 5 + HR" presents the best energy consumption for deadlines above 90 sec while visiting and deploying similar numbers of CHs and UAVs as "Algo. 5 + HA". Indeed, "Algo. 5 + HR" compensates for the prolonged data collecting time and energy (due to longer distances to CHs) by shorter flying time and less energy consumption. For deadlines below 90 sec, "Algo. 5 + HA" performs either better or similarly to "Algo. 5 + HR", in terms of energy

and number of visited CHs, since hovering exactly above CHs improves the data transmission and thus respects the deadlines. Finally, "Algo. 5 (Short) + HA" presents the worst performance as it spends more energy to reach CHs at distant locations, or in contrast, abandon them due to their data becoming outdated.

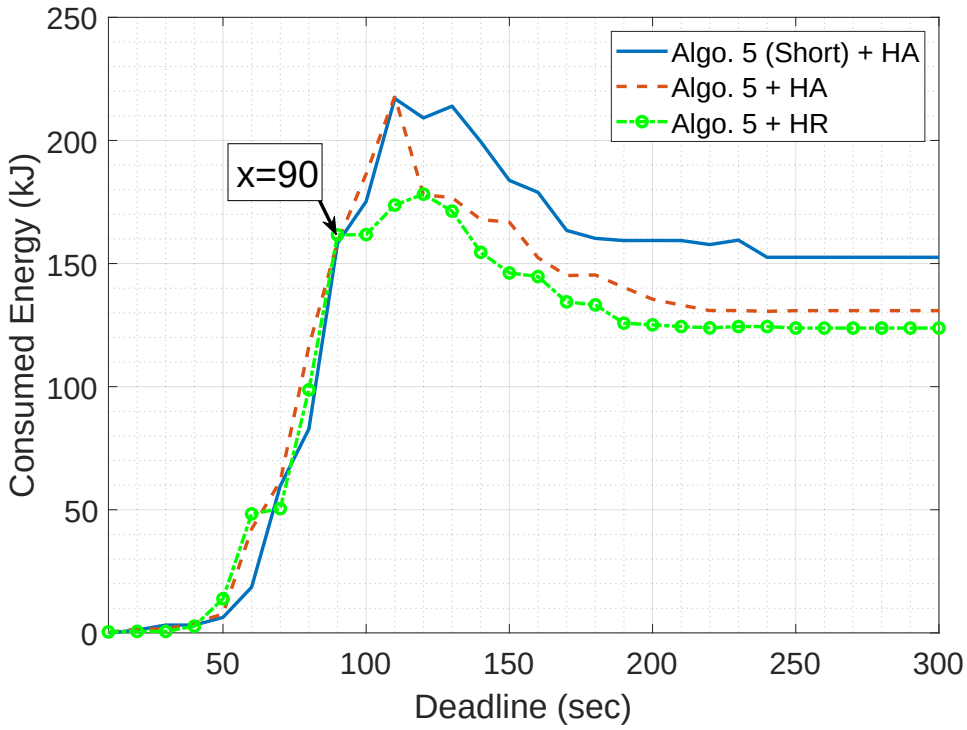


Figure 4.5: Energy consumption vs. deadline

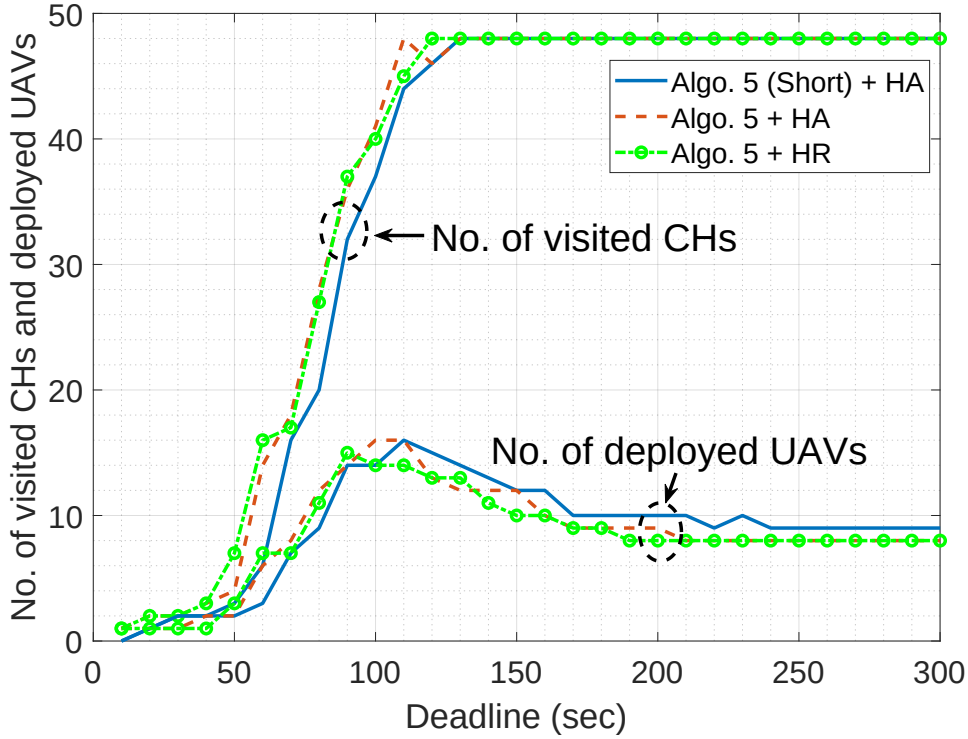


Figure 4.6: Number of CHs and UAVs vs. deadline

Fig. 4.7 investigates the impact of the environment type on the Tabu data collection approach, given the same SNs' design of Fig. 4.2. The environment parameters for eqs.(3.3)–(3.6) are given as follows: 1) Suburban (a,b)=(4.88, 0.43), 2) urban (a,b)=(9.61, 0.16), 3) dense urban (a,b)=(12.08, 0.11), and 4) highrise urban (a,b)=(27.23, 0.08) [4].

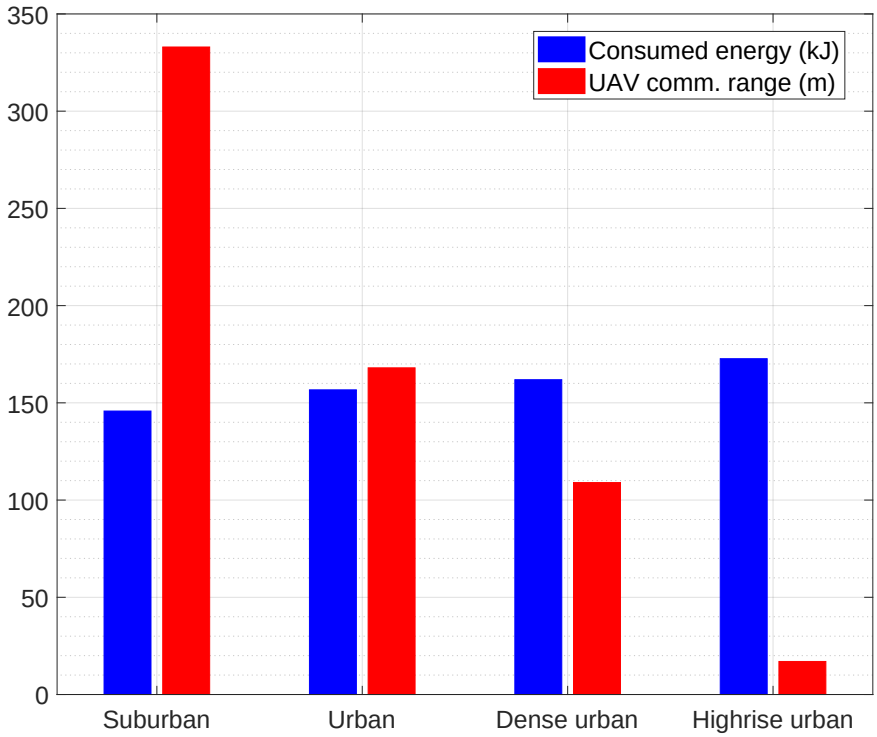


Figure 4.7: Impact of the environment type ($S_u(0)$=2500 mAh, $T_{d,c}$=150 sec)

We see that as the environment becomes denser, more energy is spent for data collection. Indeed, due to obstacles and interference, communication links are more likely to be degraded, thus causing a shorter UAV communication range. Consequently, UAVs have to fly closer to CHs in order to collect data. In contrast, in a non-obstructed environment (e.g., suburban), communication links are dominated by LoS, thus UAVs can reach CHs through short displacements.

4.3 Online path planning results

For the online approaches, we use pytorch to train a multi-task agent for 1024 episodes using replay memory and target network separation to stabilize the training process. We assume that the DRL agent employs three hidden fully-connected layers, composed of 512 neurons for the first two layers and 256 neurons for the last one. The input layer size is the same as for the state space, while the output layer size is equal to the total number of actions. Moreover, we use

rectified linear units (ReLU) as activation functions and the Adam optimizer for neural network weights update. For maximum outcome, we tune and evaluate the hyperparameters until the best performances are obtained. The latter are summarized in Table 4.2.

Table 4.2: RL hyperparameters

Data collection reward	Φ_1=100	Deadline penalty	Φ_2=-5
Moving penalty	Ψ_1=-0.05	Revisited cell penalty	Ψ_2=-0.05
Discount factor	γ=0.99	Sampling batch size	N_b=64
Learning rate	l_r=0.0001	Exploration rate	ϵ=0.01

Finally, exploration is performed in training using independent ϵ-greedy action selection by the DRL agent over its value-function $\mathcal{Q}$. Throughout the training, ϵ is linearly annealed from 1.0 to 0.01 over 500 episodes, then kept constant for the remaining episodes.

For the subsequent simulations, we assume M=1800 IoT sensors distributed within a square gridworld of size of 2000 × 2000 cells (an area of 2 × 2 km^2), and using the same simulation parameters of Table 4.1. For each episode e, we assume that the locations of SNs (and subsequently of CHs) are varying and unknown for the UAVs. The latter have to "blindly" explore the cells and reach the CHs to collect the maximum amount of data within the time and energy limitations.

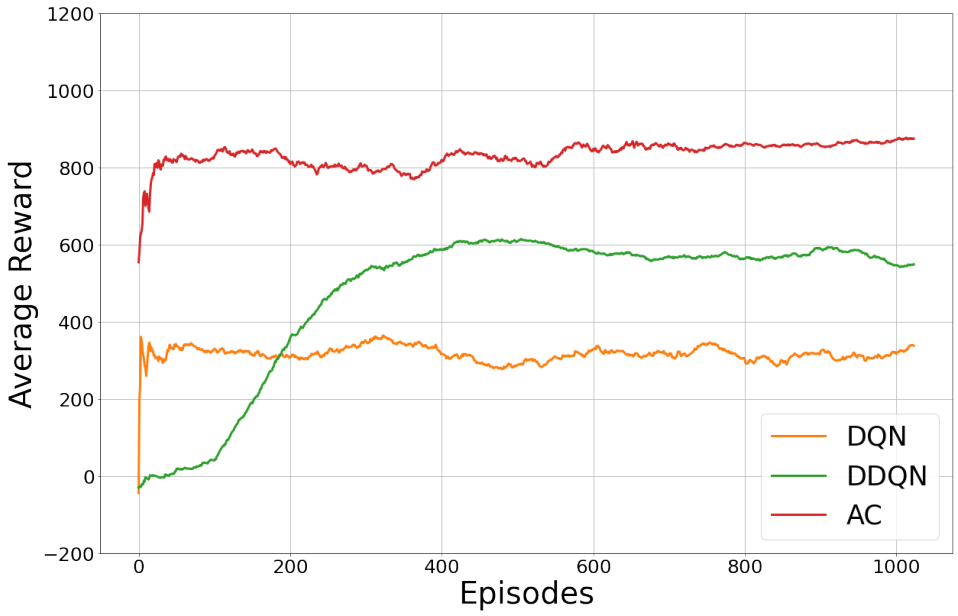


Figure 4.8: Reward vs. number of episodes using 1 UAV

In order to emphasize the efficiency of the proposed DRL algorithms, we depict in Figs. 4.8–4.9–4.10 their convergence behaviors in terms of reward (eq.(3.27)), given three deployed UAVs. As it can be seen, DQN and AC are the fastest to converge (in less than 100 episodes), while DDQN convergences after 400 episodes. The slow convergence of DDQN is due to its architectural complexity, composed of two parallel neural networks. Anyhow, AC achieves the best performance due to its superiority through the joint optimization of the value function and policy.

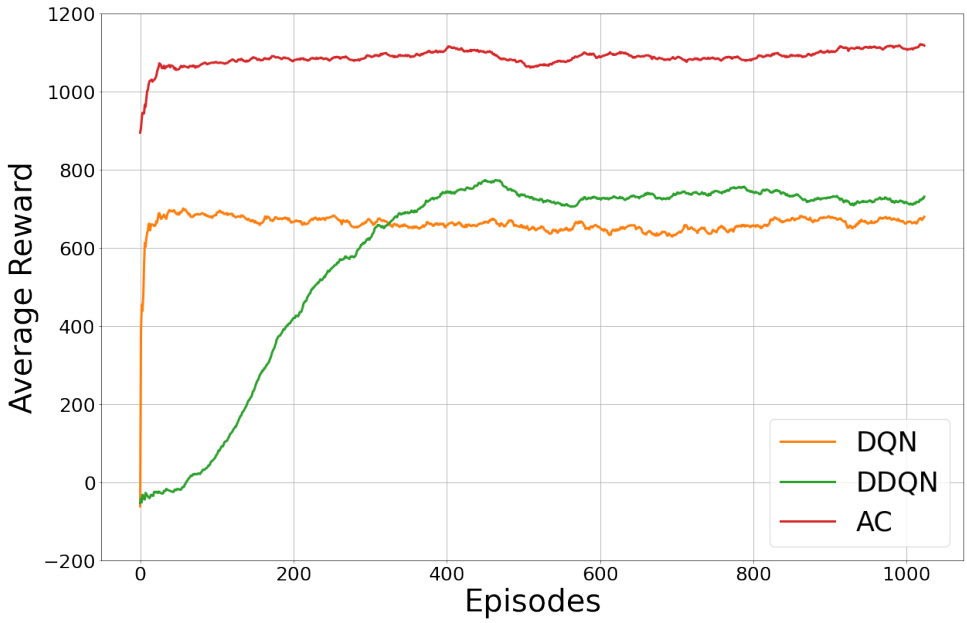


Figure 4.9: Reward vs. number of episodes using 2 UAVs

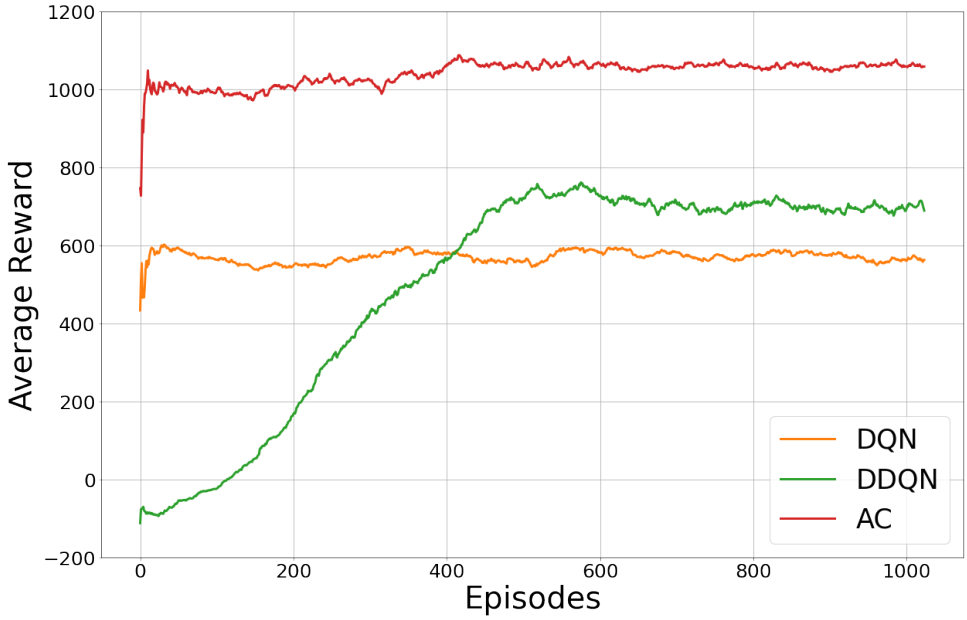


Figure 4.10: Reward vs. number of episodes using 3 UAVs

For a fair comparison between the online DRL approaches, we define the "percentage of collected data" metric, defined as the ratio of collected data from visited CHs to the total data to collect. In Fig. 4.11, we illustrate the results for different numbers of UAVs, and where the offline Tabu search performance is added as the upper-bound benchmark, called "OFF". The OFF method is able to collect most of the data (up to 100% for 2 and 3 UAVs) with respect to the deadlines since it has a full knowledge of the system status, i.e., CHs locations and deadlines. However, DQN, DDQN, and AC collect less data, e.g., using 3 UAVs, they collect 65%, 67%, and 74.5% within the deadlines respectively, but 87%, 93.5% and 94.7% when including outdated collected data. The obtained DRL results are very interesting, especially for AC, given the total absence of information about the system at the UAVs.

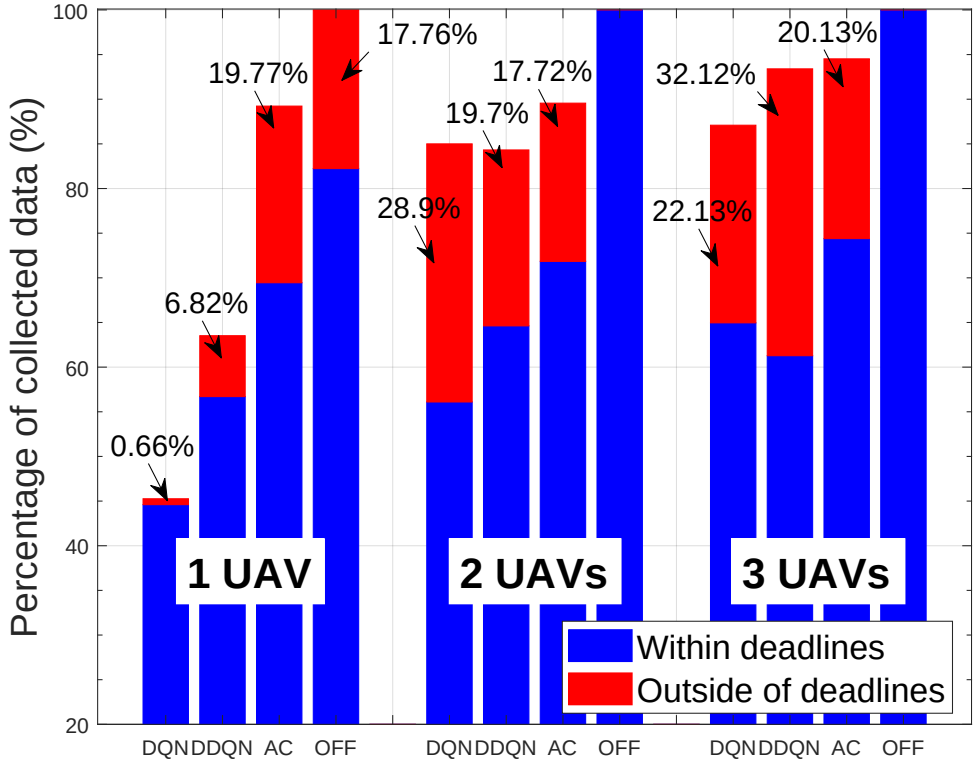


Figure 4.11: Percentage of collected data (within and outside of deadlines) vs. path planning approaches and number of UAVs

Fig. 4.12 shows the UAVs' normalized energy consumption for the path planning approaches, where the consumed energy is normalized over the percentage of visited CHs within deadlines. AC achieves the best online performance, which enhances with the number of UAVs. This is expected since AC is more efficient in selecting on-the-fly policies than DQN and DDQN.

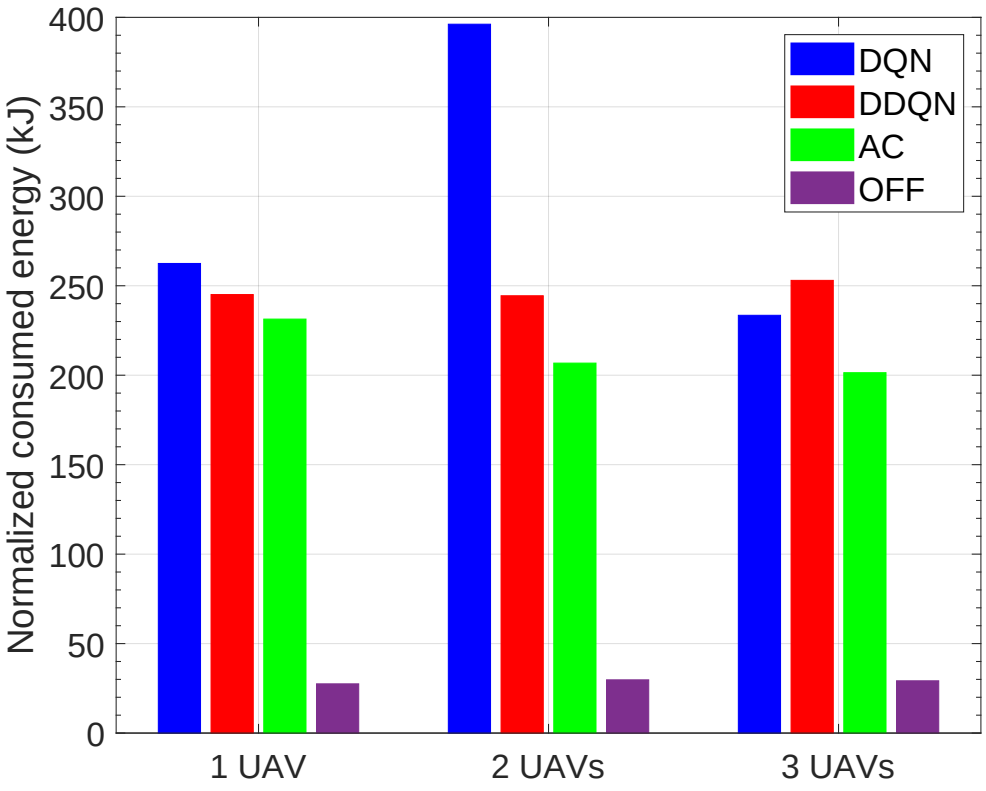


Figure 4.12: Energy consumption vs. number of UAVs

In the remaining results, we present only the performances of the OFF and AC methods, for which we evaluate the impact of the battery capacity and data collection deadline. For rapid execution and convergence, we call for transfer learning [56], which uses a pre-trained AC model in different environment conditions.

Figs. 4.13–4.14 studies the impact of the battery capacity on the performances of the system, for three deployed UAVs. In terms of the percentage of collected data within deadlines, we see that for a battery capacity less or equal to 1000 mAh, AC cannot collect more than 5% of the data. However, for a battery capacity larger than 1000 mAh, a maximum collected data of 60% is obtained and cannot be outperformed, thus drawing the limit of AC in this scenario. In contrast, OFF succeeds in collecting 100% of the data starting from a battery capacity equal to 1500 mAh. In terms of energy consumption, AC consumes a high amount of energy when the battery capacity is higher than 1000 mAh. Indeed, UAVs attempt to explore more CHs with their additional energy, however, due to the system's status lack of knowledge, they often reach CHs after the deadlines expiration. Consequently, the optimal battery capacity of the AC method is 1250 mAh, compared to 1500 mAh for the OFF method.

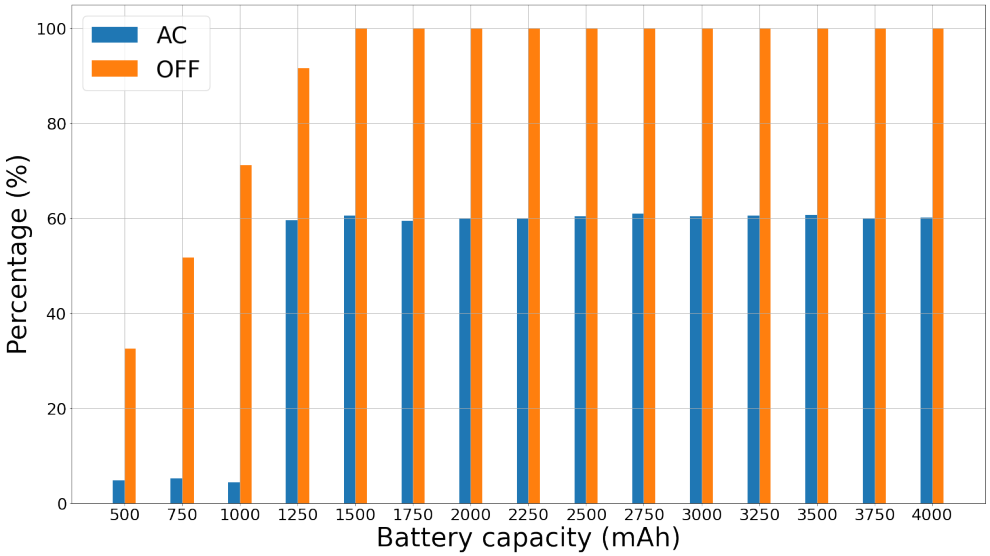


Figure 4.13: Percentage of collected data vs. battery capacity

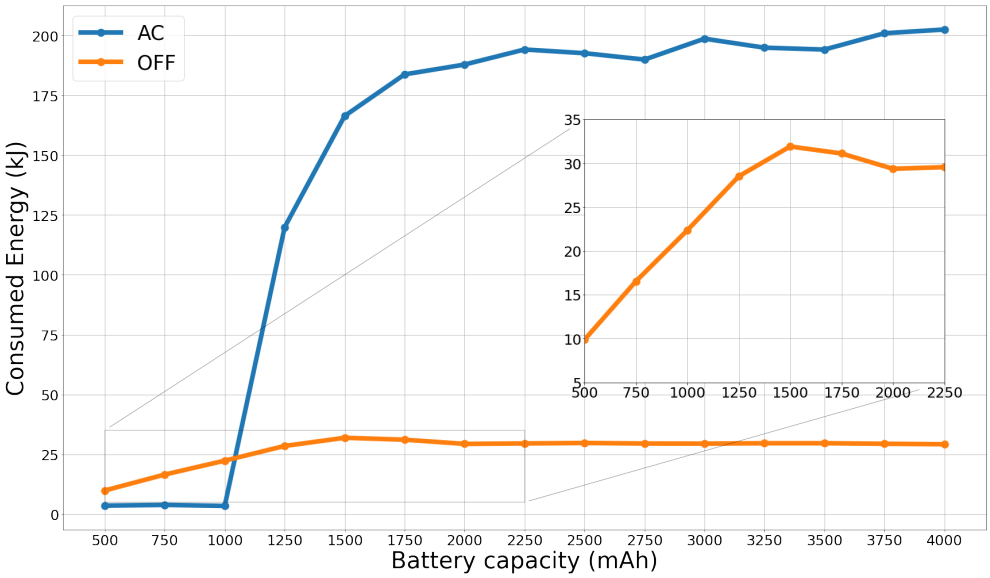


Figure 4.14: Energy consumption vs. battery capacity

In Figs. 4.15–4.16, we investigate the impact of the deadline on the performances of the system, given three deployed UAVs. In terms of percentage of collected data, as the deadline increases, both OFF and AC improve until reaching a saturation value. Indeed, OFF saturates from deadline 150 sec, for which 100% of the data is collected, while AC reaches a maximum of 90% starting from deadline 750 sec. This large gap is due to the advantageous knowledge of the system status in the OFF method. From the energy consumption perspective, OFF reaches a consumption peak at deadline 150 sec, then decreases, mainly due to better UAV paths. However, AC consumes a lot of energy for short deadlines since the UAVs waste a lot of energy exploring, without any income, their environment. Then,

it improves until reaching a stable value starting from deadline 750 sec. Indeed, given a larger deadline, the AC policy is capable of collecting more data that was previously inaccessible or reached after deadline expiration.

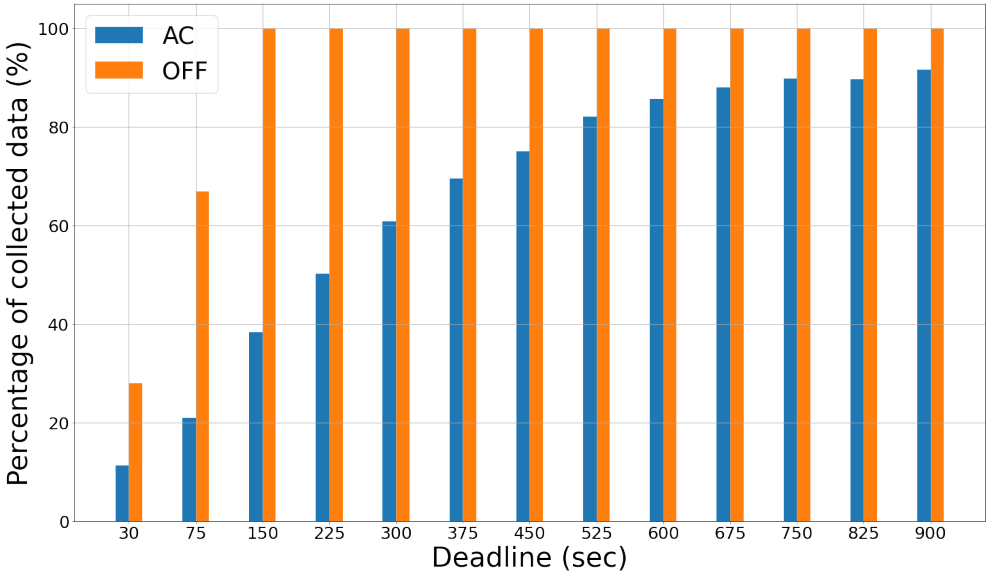


Figure 4.15: Percentage of collected data vs. deadline

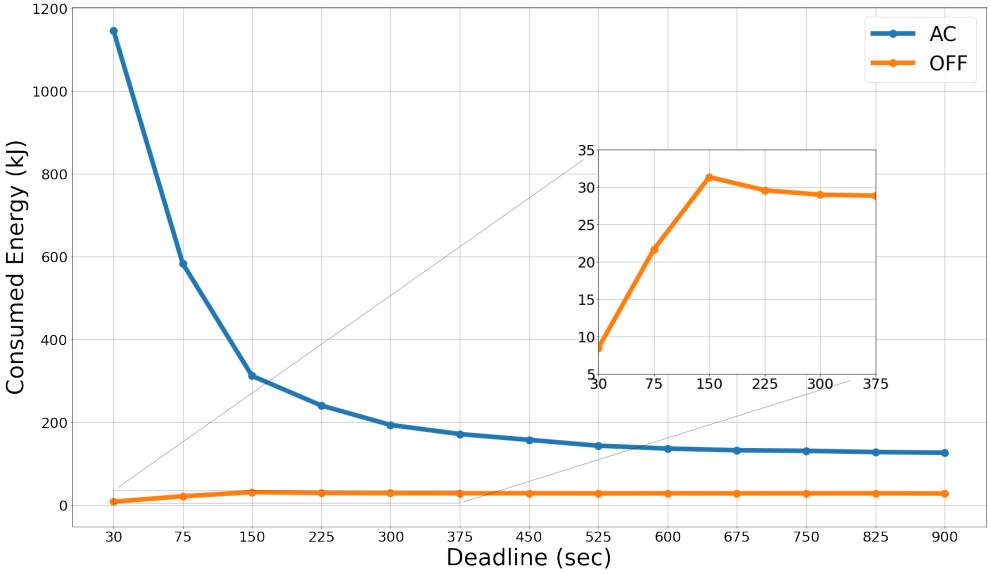


Figure 4.16: Energy consumption vs. deadline

Conclusion

In this final chapter, we presented our simulation and result for both offline and online scenarios. Simulation results show the efficiency of our proposed solution in terms of energy consumption and the amount of collected data. The next chapter will include a general conclusion and perspectives.

GENERAL CONCLUSION

In this report, we investigated the multi-UAV data collection problem in clustered IoT networks, where sensed data have time deadlines. Aiming to optimize the data collection deployment costs, we proposed a two-step solution.

In the first step, the number and locations of CHs, which gather data from associated IoT sensors, are optimized using a customized K-means approach. Simulation results demonstrated the efficiency of our proposed clustering method compared to baseline approaches. Also, moving CHs closer to the UAV dockstation provided a significant energy gain.

For the data collection step, two scenarios have been considered, namely offline and online. For the offline setting, we showed that Tabu search achieved the best performances in terms of energy consumption and number of deployed UAVs, while collecting 100% of the data. Moreover, we proved that UAV battery capacity needs to be carefully optimized, and that the deadline constraint impacts significantly the system's performances. Then, we illustrated the importance of the environment's type, where the UAVs operate. For the online setting, we found that AC outperforms DQN and DDQN in terms of amount of collected data and energy consumption. Indeed, AC is capable of collecting up to 90% of the data without any prior knowledge of the IoT network, given optimized battery capacity and sufficiently high deadlines.

In spite of the huge potential benefits of the UAV-based IoT data collection for large-scale IoT systems, there are still certain limitations that must be investigated to fully realize UAV-enabled frameworks. Energy consumption is a primary bottleneck that can limit the communication performance and endurance time of UAVs due to the limited battery capacity. In our future works, we plan to use

energy harvesting to extend flight time. The battery is recharged from green renewable energy sources such as solar, wind and electromagnetic radiations while flying without the need of landing. Moreover, our work focuses on data collection from deadline-based IoT applications which implies a real time data analytics to make decision before the information become outdated. Thus, we may adopt edge and Fog computing principles to solve some critical issues and improve performance. Edge computing offers a tolerable computational capacity, enough storage space, and fast response time to satisfy IoT application requirements. In addition, we plan to collect data from more complex use cases considering not only fixed SNs but also mobile sensors. Finally, we plan to use the capabilities of blockchain to secure all data transactions and to prevent any privacy leakage.

LIST OF PUBLICATIONS

[1] O. Ghdiri, W. Jaafar, S. Alfattani, J. B. Abderrazak and H. Yanikomeroglu, "Energy-Efficient Multi-UAV Data Collection for IoT Networks with Time Deadlines," *Proc. GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Taipei, Taiwan, 2020.

[2] O. Ghdiri, W. Jaafar, S. Alfattani, J. B. Abderrazak and H. Yanikomeroglu, "Offline and Online UAV-enabled Data Collection in Time constrained IoT Networks," *in IEEE Transactions on Green Communications and Networking - Special Issue - Selected papers from IEEE Globecom'20 (**submitted**).*

REFERENCES

[1] V. Chamola, V. Hassija, V. Gupta, and M. Guizani, "A comprehensive review of the COVID-19 pandemic and the role of IoT, drones, AI, blockchain, and 5G in managing its impact," *IEEE Access*, vol. 8, pp. 90 225–90 265, May 2020.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surv. Tuts.*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015.

[4] S. Alfattani, W. Jaafar, H. Yanikomeroglu, and A. Yongacoglu, "Multi-UAV data collection framework for wireless sensor networks," in *Proc. IEEE Glob. Comm. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[5] T. Kim and D. Qiao, "Energy-efficient data collection for IoT networks via cooperative multi-hop UAV networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13 796–13 811, Nov. 2020.

[6] M. Samir, S. Sharafeddine, C. M. Assi, T. M. Nguyen, and A. Ghrayeb, "UAV trajectory planning for data collection from time-constrained IoT devices," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 34–46, Jan. 2020.

[7] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "AoI-minimal trajectory planning and data collection in UAV-assisted wireless powered IoT networks," *IEEE Internet of Things J.*, vol. 8, no. 2, pp. 1211–1223, Jan. 2021.

[8] W. Jaafar and H. Yanikomeroglu, "Dynamics of laser-charged uavs: A battery perspective," *IEEE Internet of Things J. (Early Access)*, pp. 1–1, Dec. 2020.

[9] J. K. Lenstra and A. H. G. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, Summer 1981.

[10] M. M. Afsar and M.-H. Tayarani-N, "Clustering in sensor networks: A literature survey," *Elsevier J. Netw. Comput. Appl.*, vol. 46, pp. 198–226, Nov. 2014.

[11] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Elsevier Comput. Commun.*, vol. 30, no. 14-15, pp. 2826–2841, Oct. 2007.

[12] L. Xu, R. Collier, and G. M. O'Hare, "A survey of clustering techniques in WSNs and consideration of the challenges of applying such to 5G IoT scenarios," *IEEE Internet of Things J.*, vol. 4, no. 5, pp. 1229–1249, Oct. 2017.

[13] D. Puschmann, P. Barnaghi, and R. Tafazolli, "Adaptive clustering for dynamic IoT data streams," *IEEE Internet of Things J.*, vol. 4, no. 1, pp. 64–74, Feb. 2017.

[14] X. Shao, C. Yang, D. Chen, N. Zhao, and F. R. Yu, "Dynamic IoT device clustering and energy management with hybrid NOMA systems," *IEEE Trans. Indus. Inform.*, vol. 14, no. 10, pp. 4622–4630, Oct. 2018.

[15] A. Mukherjee, D. K. Jain, and L. Yang, "On-demand efficient clustering for next generation IoT applications: A hybrid NN approach," *IEEE Sensors J.*, Sep. 2020.

[16] A. A.-H. Hassan, W. M. Shah, A.-H. H. Habeb, M. F. I. Othman, and M. N. Al-Mhiqani, "An improved energy-efficient clustering protocol to prolong the lifetime of the WSN-based IoT," *IEEE Access*, vol. 8, pp. 200 500–200 517,

[17] C. Zhan, Y. Zeng, and R. Zhang, "Energy-efficient data collection in UAV enabled wireless sensor network," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 328–331, Jun. 2018.

[18] C. You and R. Zhang, "Hybrid offline-online design for UAV-enabled data harvesting in probabilistic LoS channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 3753–3768, Jun. 2020.

[19] Z. Wang, R. Liu, Q. Liu, J. S. Thompson, and M. Kadoch, "Energy-efficient data collection and device positioning in UAV-assisted IoT," *IEEE Internet of Things J.*, vol. 7, no. 2, pp. 1122–1139, Feb. 2020.

[20] C. Lin, G. Han, X. Qi, J. Du, T. Xu, and M. Martinez-Garcia, "Energy-optimal data collection for UAV-aided industrial WSN-based agricultural monitoring system: A clustering compressed sampling approach," *IEEE Trans. Indus. Inform. (Early Access)*, pp. 1–10, Sep. 2020.

[21] A. T. Albu-Salih and S. A. H. Seno, "Energy-efficient data gathering framework-based clustering via multiple UAVs in deadline-based WSN applications," *IEEE Access*, vol. 6, pp. 72 275–72 286, Nov. 2018.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[23] M. Yi, X. Wang, J. Liu, Y. Zhang, and B. Bai, "Deep reinforcement learning for fresh data collection in UAV-assisted IoT networks," in *Proc. IEEE Conf. Comp. Commun. Wrkshps. (INFOCOM WKSHPS)*, Aug. 2020, pp. 716–721.

[24] P. Tong, J. Liu, X. Wang, B. Bai, and H. Dai, "Deep reinforcement learning for efficient data collection in UAV-aided Internet of things," in *Proc. IEEE Int. Conf. Commun. Wrkshps. (ICC Wrkshps.)*, Jun. 2020, pp. 1–6.

[25] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "UAV path planning for wireless data harvesting: A deep reinforcement learning approach," Jul. 2020.

[26] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "Multi-UAV path planning for wireless data harvesting with deep reinforcement learning," Oct. 2020.

[27] W. Liu, P. Si, E. Sun, M. Li, C. Fang, and Y. Zhang, "Green mobility management in UAV-assisted IoT based on dueling DQN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[28] Y.-H. Hsu and R.-H. Gau, "Reinforcement learning-based collision avoidance and optimal trajectory planning in UAV communication networks," *IEEE Trans. Mob. Comput.*, Jun. 2020.

[29] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, "A UAV-assisted data collection for wireless sensor networks: Autonomous navigation and scheduling," *IEEE Access*, vol. 8, pp. 110 446–110 460, Jun. 2020.

[30] O. Ghdiri, W. Jaafar, S. Alfattani, J. B. Abderrazak, and H. Yanikomeroglu, "Energy-efficient multi-UAV data collection for IoT networks with time deadlines," in *Proc. IEEE Glob. Comm. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[31] "The internet of things: Mapping the value beyond the hype," Mckinsey Global Institute, Tech. Rep., 2015.

[32] W. Wang, Y. Xu, and M. Khanna, "A survey on the communication architectures in smart grid," *Computer Networks*, vol. 55, no. 15, pp. 3604–3629, 2011.

[33] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[34] S. Kim and S. Kim, "User preference for an iot healthcare application for lifestyle disease management," *Telecommunications Policy*, vol. 42, no. 4, pp. 304–314, 2018.

[35] L. Oliveira and J. Rodrigues, "Wireless sensor networks: A survey on environmental monitoring," *JCM*, vol. 6, pp. 143–151, Apr. 2011.

[36] P. Scully. (). "Top 10 iot applications in 2020," [Online]. Available: `https://iot-analytics.com/top-10-iot-applications-in-2020/`. (accessed: 13.12.2020).

[37] 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.

[38] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.

[39] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[40] F. Nielsen, *Ch. 8: Hierarchical Clustering - Introduction to HPC with MPI for Data Science*. Springer, 2016.

[41] D. Müllner, *Modern hierarchical, agglomerative clustering algorithms*, 2011.

[42] G. B. Dantzig and J. H. Ramser, "The Truck Dispatching Problem," *Management Science*, vol. 6, no. 1, pp. 80–91, Oct. 1959.

[43] C. Voudouris and E. Tsang, "Guided local search and its application to the traveling salesman problem," *European Journal of Operational Research*, vol. 113, no. 2, pp. 469–499, 1999.

[44] F. Glover, M. Laguna, and R. Marti, *Tabu Search*. Jul. 2008, vol. 16.

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

[46] L.-H. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3/4, pp. 69–97, 1992.

[47] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th Conf. Artificial Intelli. (AAAI)*, Phoenix, Arizona, Feb. 2016, pp. 2094–2100.

[48] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018.

[49] A. Al-Hourani, K. Sithamparanathan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.

[50] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2329–2345, Mar. 2019.

[51] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. Ann. Hawaii Int. Conf. Syst. Sci.*, Jan. 2000.

[52] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

[53] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient K-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. and Mach. Intel.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[54] D. Knuth. (). "Capacitated vehicle routing problem, OR-tools," [Online]. Available: `https://developers.google.com/optimization/routing/cvrp`. (accessed: 01.05.2020).

[55] D. C. Marinescu, "Chapter 12 - Big data, data streaming, and the mobile cloud," in *Cloud Computing*, 2nd Ed., Morgan Kaufmann, 2018, pp. 439–487.

[56] Z. Zhu, K. Lin, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *arXiv*, Sep. 2020.

Appendices

Energy-Efficient Multi-UAV Data Collection for IoT Networks with Time Deadlines

Oussama Ghdiri, Wael Jaafar, Safwan Alfattani, Jihene Ben Abderrazak, and Halim Yanikomeroglu

***Abstract*—In this paper, we focus on energy-efficient UAV-based IoT data collection in sensor networks in which the sensed data have different time deadlines. In the investigated setting, the sensors are clustered and managed by cluster heads (CHs), and multiple UAVs are used to collect data from the CHs. The formulated problem is solved through a two-step approach. In the first step, an efficient method is proposed to determine the minimal number of CHs and their best locations. Subsequently, the minimal number of UAVs and their trajectories are obtained by solving the associated capacitated vehicle routing problem. Results show the efficiency of our proposed CHs placement method compared to baseline approaches, where bringing the CHs closer to the dockstation allows significant energy savings. Moreover, among different UAV trajectory planning algorithms, Tabu search achieves the best energy consumption. Finally, the impact of the battery capacity and time deadline are investigated in terms of consumed energy, number of visited CHs, and number of deployed UAVs.**

I. Introduction

The vision of future wireless technologies with numerous smart applications and services has emerged with the Internet of Things (IoT), connecting millions of devices and people. IoT sensor networks are a key enabler for novel applications, including smart cities, connected cars, smart grids, and intelligent transportation. Indeed, the Third Generation Partnership Project (3GPP) has issued recently several standardization documents to support IoT in 5G systems (e.g., TR 23.700-20 and TR 23.700-24). Typically, IoT sensors are low-cost and limited-energy devices with short transmission ranges. In large-scale IoT systems, the deployment of super-sensors or cluster heads (CHs) to aggregate data from neighboring sensors is a practical solution to reduce communication overhead and save sensors' energy. However, even CHs may have a limited communication range, and therefore, a data collector needs to be deployed to gather sensed data and send it to the control center. In that matter, UAVs can be deployed as data collectors [1]. Indeed, UAVs have been presented as an efficient tool to solve several wireless communication challenges [2]–[4]. Accordingly, recent 3GPP standards were issued to regulate different aspects of UAV support in 5G (TR 22.829 and TS 22.125). Due to the agility, strong Line-of-Sight (LoS), and fast deployment of UAVs, the latter are a relevant option for data collection in IoT systems [1]. Nevertheless, UAVs have a limited onboard energy and are sensitive to blockages. Hence, accurate communication channel and power consumption models should be considered when designing data collection missions. On the other hand, large-scale IoT networks encompass sensors with different priorities, criticality levels, and time-sensitive usefulness. Consequently, energy-efficient data collection needs an accurate UAV deployment and trajectory planning to meet time-sensitive constraints, e.g., time deadlines for data collection.

Several works proposed UAV-based sensors' data collection. In our previous work [1], we proposed collecting data from a clustered wireless sensor network using one or several UAVs, aiming to minimize the mission costs in terms of number of clusters and traveled distance by UAVs, while guaranteeing data is collected from all clusters. In this setting, sensor nodes (SNs) were clustered using a K-means-based algorithm, and UAV trajectories designed with different approaches, including genetic algorithm and nearest-neighbor. In [5], the problem of minimizing data collecting time from IoT time-constrained sensors is investigated, where the trajectory of a single collecting UAV and radio resource allocation are optimized. Moreover, the authors of [6] proposed joint optimization of wakeup schedule of SNs and a single UAV's trajectory to minimize the total consumed energy by SNs, while guaranteeing complete data collection from all of them. Finally, authors in [7] presented a data collection framework, where multiple UAVs are deployed to collect data from clustered SNs, aiming to minimize the number of deployed UAVs and their flight times and distances, with respect to a mission deadline. The optimal solution is obtained, which outperforms a benchmark greedy approach. Nevertheless, these works have several limitations. Authors in [1], [5] did not consider energy restrictions of UAVs, while [6] focused only on single UAV trajectory planning. Finally, [7] did not account for energy limitations and air-to-ground accurate channel modeling.

Motivated by the aforementioned points, we propose here a practical data collection approach for large-scale IoT systems, where multiple UAVs collect sensed data with different time deadlines. The proposed solution follows a two-step approach. First, the minimum number of CHs and their locations are determined to guarantee that data from all SNs is aggregated at the CHs. Then, the number of collecting UAVs and their trajectories are optimized to minimize the total consumed energy. Finally, the impact of key parameters, e.g., UAV's battery capacity and data time deadlines, is investigated.

O. Ghdiri and J. Ben Abderrazak are with ESPRIT School of Engineering Tunis, Tunisia, W. Jaafar and H. Yanikomeroglu are with Carleton University, Canada, and S. Alfattani is with the University of Ottawa, Canada, and King Abdulaziz University, Saudi Arabia.

The authors would like to thank Professor Abbas Yongacoglu, University of Ottawa, Canada, for his comments about this work.

II. System Model

A. Network and Channel Models

We consider an IoT sensor network consisting of a set $\mathcal{M} = \{1, \ldots, M\}$ of SNs, randomly located in a large area and constantly collecting time-sensitive data. We assume that SNs are heterogeneous, i.e., sensed data by SNs are of different kind, having different importance in time. Indeed, some sensed data are more critical and need to be collected faster than other less-critical data. Due to the limited energy and communication capability of typical IoT sensors, we assume that a set of $\mathcal{K} = \{1, \ldots, K\}$ cluster heads (CHs) can be deployed to collect, aggregate, and transmit sensed data to the data collector. In our system, collection of data is operated by multiple UAVs, which fly and hover above/close to CHs for a sufficient time to collect the aggregated data.

Each sensor transmits its sensed data to its associated CH using power P_{SN}, while a CH communicates with the collecting UAV using power $P_{\text{CH}} > P_{\text{SN}}$. Assuming that IoT sensors communicate with their associated CH using the time-division multiple access (TDMA) protocol, the received signal-to-noise ratio (SNR) at CH c for the signal transmitted by SN s, denoted γ_{sc}, can be written as

$$\gamma_{sc} = \frac{P_{\text{SN}} d_{sc}^{-\alpha}}{\sigma^2}, \forall s \in \mathcal{M},\ \forall c \in \mathcal{K} \tag{1}$$

where $d_{sc} = ||\mathbf{q}_s - \mathbf{q}_c||$ is the distance between IoT sensor s and CH c, $\mathbf{q}_i = [x_i, y_i, z_i]$ is the 3D location of node i ($i = s$ or c), α is the path loss exponent, and σ^2 is the noise power. This communication link is considered successful if $\gamma_{sc} \geq \gamma_{\text{th}}$, where γ_{th} is a selected SNR threshold. Hence, a maximum communication range for each SN s associated with CH c is defined as [1]

$$d_{sc} \leq d_{\text{th}} = \left(\frac{P_{\text{SN}}}{\sigma^2 \gamma_{\text{th}}}\right)^{1/\alpha}, \forall s \in \mathcal{A}_c,\ c \in \mathcal{K} \tag{2}$$

where $\mathcal{A}_c$ is the set of SNs associated with CH c.

Given (2), CHs are deployed to collect data from SNs and send it to UAVs. Let $\mathcal{U} = \{1, \ldots, U\}$ be the set of available UAVs for CHs data collection. For the sake of simplicity, we assume that a UAV flies at a fixed altitude H above ground, such that H respects authority regulations and safety considerations. Also, UAVs are equipped with sensors that allow obstacle avoidance at a certain safety distance, denoted d_{safe}, when they fly at maximum speed $v_{\max}$.

When UAV u hovers to receive data from CH c, the air-to-ground channel between them can be expressed using the probabilistic path loss model given by [8]

$$\Lambda_{cu} = Pr_{cu}^{\text{LoS}} l_{cu}^{\text{LoS}} + Pr_{cu}^{\text{NLoS}} l_{cu}^{\text{NLoS}},\ \forall c \in \mathcal{K}_u,\ \forall u \in \mathcal{U}, \tag{3}$$

where Λ_{cu} is the average path-loss between CH c and UAV u, $\mathcal{K}_u \subset \mathcal{K}$ is the subset of ordered CHs to visit by UAV u, Pr_{cu}^{LoS} is the LoS probability, $Pr_{cu}^{\text{NLoS}} = 1 - Pr_{cu}^{\text{LoS}}$ is the NLoS probability, and l_{cu}^{LoS} and l_{cu}^{NLoS} are LoS and NLoS path-loss, respectively. These parameters are written as [8]

$$Pr_{cu}^{\text{LoS}} = 1/\Big(1 + ae^{b\ (\theta_{cu} - a)}\Big),\ \forall c \in \mathcal{K}_u,\ \forall u \in \mathcal{U}, \tag{4}$$

and

$$l_{cu}^{m} = 20\log(4\pi f_c/V) + 20\log(d_{iu}) + \beta_m,\ m = \text{LoS or NLoS} \tag{5}$$

where a and b are constant values determined from the environment (rural, urban, etc.), $\theta_{cu} = \frac{180}{\pi} \times \arcsin\left(\frac{H}{d_{cu}}\right)$ is the elevation angle between UAV u and CH c, f_c is the carrier frequency, V is the speed of light, and β_m is the excessive path-loss coefficient. Using Friis formula, the received power at UAV u from CH c is expressed by

$$P_{cu} = P_{\text{CH}} - \Lambda_{cu},\ \forall c \in \mathcal{K}_u,\ \forall u \in \mathcal{U}, \tag{6}$$

where P_{CH} is the unit power (in Watt), i.e. equal to 0 dBm.

B. UAV Data Collection Model

We assume that the mission time, i.e., the maximum time for UAVs to collect data from CHs and return to their dockstation, is T_F and divided into N time slots (TSs). Thus, TSs are equal and are of length $\delta = \frac{T_F}{N}$. We define the 3D location of UAV u in time slot t as $\mathbf{q}_u^t = [x_u^t, y_u^t, H]$. Since the UAV's speed is limited by $v_{\max}$, its traveled distance in one TS is constrained by

$$||\mathbf{q}_u^{t+1} - \mathbf{q}_u^t|| = \sqrt{(x_u^{t+1} - x_u^t)^2 + (y_u^{t+1} - y_u^t)^2} \leq v_{\max}\delta. \tag{7}$$

In our system, each CH c uploads its data to its associated UAV u at rate R_{cu} in bits/second (bps), determined using the Shannon equation as follows:

$$R_{cu} = W\log_2(1 + \gamma_{cu}),\ \forall c \in \mathcal{K}_u,\ \forall u \in \mathcal{U}, \tag{8}$$

where W is the total bandwidth of the channel and $\gamma_{cu} = \frac{\bar{P}_{cu}}{\sigma^2}$ is the SNR, where $\bar{P}_{cu} = 10^{\frac{P_{cu}}{10}}$ is the linear value of P_{cu}. Accordingly, the required number of TSs to upload one packet to the UAV can be expressed by

$$T_{1,cu} = \left\lceil \frac{S_p}{R_{cu}\delta} \right\rceil,\ \forall c \in \mathcal{K}_u,\ \forall u \in \mathcal{U}, \tag{9}$$

where S_p is the size of the packet in bits and $\lceil \cdot \rceil$ is the ceiling function. Given Q_c data packets to be transmitted by CH c, then the required data collection time (in TSs) by UAV c is

$$T_{cu} = Q_c T_{1,cu}\ \forall c \in \mathcal{K}_u,\ \forall u \in \mathcal{U}. \tag{10}$$

Due to the different priorities of collected data at each CH, we define by $T_{d,c}$ the time deadline of data stored in CH c, i.e., if not totally collected within this deadline, the data present in the CH becomes outdated and is dropped[1]. Hence, data collection has to respect the following constraint, assuming that all packets in one CH are collected by only one UAV

$$\sum_{m=1}^{\phi_u(c)} [T_{mu} + T_{f,u}(m-1, m)] \leq T_{d,c},\ \forall c \in \mathcal{K}_u, \tag{11}$$

where $\phi_u(c)$ designates the rank (i.e. index) of CH c in $\mathcal{K}_u$, T_{mu} is the data collection time of CH ranked m in $\mathcal{K}_u$, and $T_{f,u}(m-1, m)$ is the flight time (in TSs) from the hovering location to collect data from CH ranked $(m-1)$ to that associated to the CH ranked m in $\mathcal{K}_u$, with $m = 0$

[1]Typically, this deadline is determined by the most critical sensed data.

designating the dockstation. The flight time of UAV u between two locations $\mathbf{q}_{u,i}$ and $\mathbf{q}_{u,i'}$, where i and i' are two CHs ranks in the UAV path, can be calculated as follows:

$$T_{f,u}(i,i') = \frac{||\mathbf{q}_{u,i'} - \mathbf{q}_{u,i}||}{v_u\ \delta}, \tag{12}$$

with $v_u \leq v_{\max}$ is the flying speed of the UAV (in m/sec).

C. UAV Energy Model

The UAV is energy-constrained due to limited on-board battery. The battery lifetime depends on several factors, e.g., UAV's energy source, type, weight, speed, etc. Typically, the UAV's energy consumption consists of the propulsion energy and the communication energy. Without loss of generality, communication energy is several orders of magnitude smaller than propulsion energy. Hence, it is neglected in the considered energy model. For the propulsion energy, we adopt the propulsion-power model for rotary-wing UAVs of [9]

$$\begin{aligned} P_{\text{prop,u}}(v_u) = \zeta_I \left(\sqrt{1 + \frac{v_u^4}{4v_0^4}} - \frac{v_u^2}{2v_0^2} \right)^{1/2} \\ + \zeta_B \left(1 + \frac{3v_u^2}{U_{\text{tip}}^2} \right) + \frac{1}{2} d_0 \psi r A v_u^3, \end{aligned} \tag{13}$$

where ζ_B and ζ_I are the blade profile power and induced power, respectively. U_{tip} is the tip speed of the rotor blade, v_0 is the mean rotor induced velocity, ψ is the air density, d_0 is the fuselage drag ratio, r is the rotor solidity, and A denotes the rotor disc area. In order to obtain the power consumption for hovering, we use the same equation above, but with null flying speed $v_u = 0$, thus, we get

$$P_{\text{hov,u}} = P_{\text{prop,u}}(v_u = 0) = \zeta_B + \zeta_I. \tag{14}$$

Using (13)–(14), the consumed energy of UAV u during a time period T can be given by

$$E_u(v,T) = \begin{cases} P_{\text{prop,u}}(v) \times T, & \text{if } v > 0 \\ P_{\text{hov,u}} \times T, & \text{if } v = 0. \end{cases} \tag{15}$$

Hence, the related battery status at TS t, denoted $S_u(t)$, can be expressed by

$$S_u(t) = S_u(t-1) - E_u(v,\delta),\ \forall t > 1, \tag{16}$$

where $S_u(t-1)$ is the battery's status at the end of TS $(t-1)$, and $S_u(0)$ is the initial battery capacity. The latter is expressed as $S_u(0) = S_{\text{ini}} + S_{\min}$, where S_{ini} is the battery capacity dedicated for the mission, while $S_{\min}$ is a safety capacity, reserved for emergency pull back to the dockstation. Hence, $S_u(t) \in [S_{\min}, S_u(0)]$.

III. Problem Formulation

In this section, we formulate our optimization problem aiming to minimize the total consumed energy for data collection, by optimizing the deployment of CHs, the number of required UAVs, and their trajectories. Let $K_u = |\mathcal{K}_u|$ be the cardinality of the set $\mathcal{K}_u$, $\forall u \in \mathcal{U}$. Then, using (15), the consumed energy by UAV u during the mission is given by

$$E_u(\mathcal{K}_u) = \sum_{m=1}^{K_u+1} \left[E_u(0, T_{mu}\ \delta) + E_u(v_u, T_{f,u}(m-1,m)\ \delta) \right], \tag{17}$$

where $T_{f,u}(K_u, K_u+1) = \|\mathbf{q}_{u,K_u} - \mathbf{q}^0\|/v_u\delta$, and $\mathbf{q}^0$ is the initial location of all UAVs, i.e., the dockstation. Therefore, the optimization problem can be formulated as follows:

$$\min_{\substack{\mathcal{K},\mathcal{L},\{\mathcal{A}_c\}_{c\in\mathcal{K}} \\ \mathcal{U},\{\mathcal{K}_u,\mathcal{L}_u\}_{u\in\mathcal{U}}}} \quad \sum_{u=1}^{U} E_u(\mathcal{K}_u) \tag{P1}$$

$$\text{s.t.} \quad d_{sc} \leq d_{\text{th}},\ \forall s \in \mathcal{A}_c, \forall c \in \mathcal{K}, \tag{P1.a}$$

$$S_u(t) \geq S_{\min},\ \forall u \in \mathcal{U}, \forall 1 \leq t \leq T_u,\ \forall u \in \mathcal{U} \tag{P1.b}$$

$$\sum_{m=0}^{\phi_u(c)} (T_{mu} + T_{f,u}(m-1,m)) \leq T_{d,c}, \tag{P1.c}$$

$$\|\mathbf{q}_u^t - \mathbf{q}_{u'}^t\| \geq d_{\text{safe}}, \forall t \geq 1, \forall (u,u') \in \mathcal{U}^2,\ u \neq u' \tag{P1.d}$$

$$\|\mathbf{q}_u^{t+1} - \mathbf{q}_u^t\| \leq v_{max}\delta,\ \forall 1 \leq t \leq T_u,\ \forall u \in \mathcal{U}, \tag{P1.e}$$

where $T_u \leq N$ is the effective mission completion time for UAV u (in TSs), $\mathcal{L} = \{\mathbf{q}_c\}_{c\in\mathcal{K}}$ is the set of selected locations for CHs, and $\mathcal{L}_u = \{\mathbf{q}_{u,i}\}_{i\in\mathcal{K}_u}$ is the set of ordered UAV hovering locations to collect data from its associated CHs.

The objective function is the total consumed energy by all UAVs, given their trajectories $\mathcal{L}_u$ associated to their sets of CHs $\mathcal{K}_u$. (P1.a) ensures the successful communication between SNs in $\mathcal{A}_c$ and their associated CH c. (P1.b) guarantees that enough energy is available in the battery to complete the mission at any TS t, while (P1.c) satisfies the time deadline condition when a UAV collects data from CHs. Also, (P1.d) ensures that no collisions between UAVs occur, and finally, (P1.e) limits the flying distance, for a given $v_{\max}$.

Problem (P1) is NP-hard. Indeed, in the special case of already deployed CHs, the problem becomes finding the best routes for UAVs, while respecting the energy and time deadlines conditions. The latter can be seen as the capacitated vehicle routing problem with time windows (CVRPTW) [10]. The CVRPTW can be described as selecting the routes for a number of vehicles, aiming to serve a group of customers within time windows. Each vehicle has a limited capacity, which is used to depart from a depot point, serve a number of customers along its route, then return to the same depot point. The objective of the CVRPTW is to minimize the total transport costs. Logically, the vehicles, customers, and transport costs, can be assimilated by the UAVs, CHs, and energy consumption, respectively. Since the CVRPTW is known to be NP-hard [10], then by restriction, (P1) is NP-hard.

IV. Solution Approach

In (P1), we notice that the optimization of parameters $\mathcal{U}$ and $\{\mathcal{K}_u, \mathcal{L}_u\}_{u\in\mathcal{U}}$ directly depends on the selected parameters $\mathcal{K}, \{\mathcal{A}_c\}_{c\in\mathcal{K}}$ and $\mathcal{L}$. Moreover, since their associated constraints are independent, problem (P1) can be divided into two cascaded problems as follows:

1) A first problem of sensors clustering and CHs placement, where only $\mathcal{K}$, $\mathcal{L}$, and $\{\mathcal{A}_c\}_{c\in\mathcal{K}}$ are optimized, aiming to minimize the number of deployed CHs is formulated.[2]
2) Then, given $\mathcal{K}$, $\mathcal{L}$ and $\{\mathcal{A}_c\}_{c\in\mathcal{K}}$, a second problem of UAV trajectory planning is formulated in order to minimize the total consumed energy, through the optimization of $\mathcal{U}$ and $\{\mathcal{K}_u, \mathcal{L}_u\}_{u\in\mathcal{U}}$.

Hence, we present next the two described problems and propose efficient approaches to solve them.

A. IoT Sensors Clustering

Typically, IoT sensors are deployed in large areas in order to sense, process, and communicate relevant data for some applications. Due to their limited battery, IoT sensors need to carefully use their energy, while prolonging the life of collected data as much as possible. To do so, IoT sensors can be grouped into disjoint and non-overlapping clusters to reduce the amount of used energy. SNs scan the surrounding environment and transmit sensed data to CHs, which aggregate and transmit obtained information to the UAVs [11].

Several state-of-the-art clustering techniques exist, e.g., K-means [12], density-based spatial clustering, and hierarchical cluster analysis (HCA) [13]. Nevertheless, due to the low-complexity of K-means, we focus here on its customization, aiming to group SNs and deploy the minimal number of CHs. Thus, the associated clustering problem can be written as

$$\min_{\substack{\mathcal{K},\mathcal{L},\\ \{\mathcal{A}_c\}_{c\in\mathcal{K}}}} \quad \sum_{c=1}^{K}\sum_{s=1}^{M} a_{sc}\, d_{sc}^2 \tag{P2}$$

$$\text{s.t.} \quad d_{sc} \le d_{\text{th}}, \forall s \in \mathcal{A}_c, \forall c \in \mathcal{K} \tag{P2.a}$$

$$|\mathcal{A}_c| \le F, \quad \forall c \in \mathcal{K}, \tag{P2.b}$$

where $a_{sc} \in \{0,1\}$ is a binary variable indicating whether SN s is associated with CH c or not, $\forall s \in \mathcal{A}_c$. Constraint (P2.b) guarantees fairness in associating SNs with CHs, where F is the maximum number of SNs associated with one CH. Finally, $|\mathcal{A}_c|$ is the cardinality of the set $\mathcal{A}_c$.

Conventionally, K-means clustering requires a predefined number of CHs. Hence, in [1], the authors proposed a K-means algorithm to solve (P2) without any constraint, then reexecuted it iteratively until (P2.a) is met. Such method is time consuming and inaccurate, as the clustering performance may vary with the algorithm initialization. Therefore, we propose here to improve the method of [1] by integrating both constraints (P2.a)–(P2.b) into the clustering process. The proposed approach is presented in Algorithm 1, and described as follows. First, locations of K CHs are randomly initialized. Then, each SN is assigned to the closest CH with respect to (P2.a)–(P2.b). Next, the locations of CHs are updated by calculating the resulting mean location of associated SNs for each CH. This procedure is repeated until convergence, i.e., the calculated locations remain unchanged. Since we aim to deploy the minimal number of CHs, we execute the aforementioned steps for an increasing number of CHs until a solution to (P2) is obtained, i.e., K is determined.

Although the calculated CHs locations are an adequate solution for (P2), this may not be the case for the main problem (P1). In this context, we propose to improve the locations of CHs by making them closer to the dockstation. Indeed, CH c has a mobility margin if its furthest associated SN, denoted s_0, respects (P2.a) loosely, i.e., $d_{s_0c} < d_{\text{th}}$. Hence, making the CHs closer to the dockstation would increase the probability to respect data collection deadlines, shorten the mission time, and improve the energy consumption of UAVs. This procedure is in lines 32-34 of Algorithm 1.

Algorithm 1 IoT sensors clustering algorithm

1: **Input:** $\mathcal{M}$, $\{\mathbf{q}_s\}_{s\in\mathcal{M}}$, and $\max_{\text{it}}$ %$\max_{\text{it}}$ is the max. number of CHs
2: **Output:** Optimal $\mathcal{K}$, $\mathcal{L}$, and $\{\mathcal{A}_c\}_{c\in\mathcal{K}}$
3: Set $k = (|\mathcal{M}| \div F)$
4: **for** k **to** $\max_{\text{it}}$ **do**
5: Set $\mathcal{L}$ randomly %Initial locations of CHs
6: Set $\mathcal{L}_{old}$ %Set of zeros
7: Set $\mathcal{A}_c = \emptyset$, $\forall c = 1, \ldots, k$ %c indicates the rank of
8: CH in $\mathcal{L}$
9: Set $error = dist(\mathcal{L}, \mathcal{L}_{old})$ %Convergence parameter
10: **while** $error \neq 0$ **do**
11: **for** $s \in \mathcal{M}$ **do**
12: Calculate d_{sc}, $\forall c = 1, \ldots, k$
13: Find $c_0 = \arg(\min(d_{sc}))$
14: **if** $d_{sc_0} \le d_{\text{th}}$ & $|\mathcal{A}_{c_0} \cup \{s\}| \le F$ **then**
15: $\mathcal{A}_{c_0} = \mathcal{A}_{c_0} \cup \{s\}$ %Associate SN s with
16: CH c_0
17: **else**
18: $\mathcal{A}_{k+1} = \mathcal{A}_{k+1} \cup \{s\}$ %Put c in set of no
19: association
20: **end if**
21: **end for**
22: Set $\mathcal{L}_{old} = \mathcal{L}$
23: Calculate $\mathcal{L}$ as the mean location of the associated
24: SNs in $\mathcal{A}_c$, $\forall c = 1, \ldots, k$
25: Calculate $error = dist(\mathcal{L}, \mathcal{L}_{old})$
26: **end while**
27: **if** $\mathcal{A}_{k+1} = \emptyset$ **then**
28: $\mathcal{K} = \{1, \ldots, k\}$
29: Break
30: **end if**
31: **end for**
32: **while** $\max(d_{sc}) < d_{\text{th}}$, $\forall s \in \mathcal{A}_c$ **do**
33: Get $\mathcal{L}$ closer to the dockstation
34: **end while**

B. Multi-UAV Trajectory Planning

Given $\mathcal{K}$, $\mathcal{L}$, and $\mathcal{A}_c$, $\forall c \in \mathcal{K}$, the trajectory planning problem can be formulated as

$$\min_{\mathcal{U},\{\mathcal{L}_u,\mathcal{K}_u\}_{u\in\mathcal{U}}} \quad \sum_{u=1}^{U} E_u(\mathcal{K}_u) \tag{P3}$$

$$\text{s.t.} \quad \text{(P1.b)} - \text{(P1.e)}.$$

As discussed previously, this problem is NP-hard, and can be assimilated to a CVRPTW problem. In order to solve it, we model our system as a graph, described as follows. Let $G = (\mathcal{D}, \mathcal{E})$ be a complete graph, where $\mathcal{D} = \{0, 1, \ldots, K\}$ is a set of vertices (nodes) representing the dockstation (node

[2]This objective has a direct impact on the consumed energy of data collection since a lower number of CHs would reduce the mission time and energy consumption of UAVs.

0) and K CHs, and $\mathcal{E}$ is the set of directed edges connecting the nodes. A directed edge from node i to node j, denoted e_{ij}, represents the UAV's flying operation from i to j and the hovering operation at j. A cost associated to the edge e_{ij}, called $\chi_u(e_{ij}) = E_u(v_u, \mu_1(e_{ij})) + E_u(0, \mu_2(e_{ij}))$, is expressed as the sum of the UAV's flying and hovering energy, where $\mu_1(e_{ij}) = T_{f,u}(i,j)\delta$ and $\mu_2(e_{ij}) = T_{ju}\delta$ respectively, $\forall u \in \mathcal{U}$.[3] Moreover, the time deadlines at CHs are represented in the graph by a time window $\omega_c = [o_c, v_c]$, where o_c and v_c are the minimum and maximum instants for data collection, for each node $c \in \mathcal{D}$. This time window defines when data collection at node c can begin and end. Finally, trajectory of UAV u in $\mathcal{G}$ can be defined by $\mathcal{K}_u$, where each element of $\mathcal{K}_u$ is the ordered node to visit. Accordingly, (P3) can be reformulated as problem (P4) detailed below:

$$\min_{\substack{\mathcal{U},\{\mathcal{K}_u,\\ \mathcal{L}_u\}_{u\in\mathcal{U}}}} \quad \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{D}}\sum_{j\in\mathcal{D}} \chi_u(e_{ij})\, b_{u,ij} \tag{P4}$$

$$\text{s.t.} \quad \sum_{u=1}^{U}\sum_{i\in\mathcal{D}} b_{u,ij} \leq 1, \quad \forall j \in \mathcal{D} \tag{P4.a}$$

$$\sum_{j\in\mathcal{D}} b_{u,0j} = \sum_{i\in\mathcal{D}} b_{u,i0} = 1, \quad \forall u \in \mathcal{U} \tag{P4.b}$$

$$\sum_{i=1}^{j-1} \mu_1(e_{i(i+1)}) + \mu_2(e_{i(i+1)}) \in [o_c, v_c], \ \forall j \in \mathcal{K}_u \tag{P4.c}$$

$$\sum_{i\in\mathcal{D}}\sum_{j\in\mathcal{D}} \chi_u(e_{ij}) b_{u,ij} \leq S_u(0) - S_{\min}, \ \forall u \in \mathcal{U}, \tag{P4.d}$$

where $b_{u,ij}$ is the binary variable indicating the selection of the edge $i-j$ within UAV u's trajectory. Constraint (P4.a) ensures that each CH is visited at most once by exactly one UAV. (P4.b) guarantees that each UAV departs and returns to the dockstation. Condition (P4.c) emphasizes that the data collection time cannot exceed the time deadline. Finally, (P4.d) guarantees that the consumed energy by any UAV respects the battery capacity. Subsequently, problem (P4) can be solved using any of the available CVRPTW heuristic or metaheuristic approaches, such as gradient descent, simulated annealing, Tabu search, etc. [14].

V. Numerical Results

We assume M=2000 IoT sensors randomly and uniformly distributed within a geographical area of 5×5 km^2. Unless specified otherwise, we use the simulation parameters of Table I, as presented in the next page.

Fig. 1 presents the clustering performance of several algorithms, expressed by the average number of clusters as a function of communication range, d_{th} (averaging over 100 scenarios of SNs). We consider three clustering algorithms, namely proposed Algo. 1, K-means-based clustering in [1] (K-means Algo. [1]), and HCA-based clustering (HCA [13]). As d_{th} increases, a smaller number of CHs is needed, as the latter can be reached by a higher number of SNs. Moreover, Algo. 1 achieves the best clustering performance. Indeed, unlike the other algorithms, Algo. 1 is capable of adjusting the clusters to constraints (P2.a)–(P2.b) on-the-fly, i.e., while processing the SNs.

In Fig. 2, we depict a scenario where Algo. 1 is used to group 2000 SNs into 48 clusters, from which data is collected using 11 UAVs. We distinguish two variations, Algo. 1 as is, and "Algo. 1 (Short)", where lines 32–34 are omitted. It is obvious that Algo. 1 would result in a more energy-efficient data collection, since UAVs would fly for less time, assuming that they hover directly above each CH to collect data.

For the clustering design of Fig. 2, Figs. 3–4 compare the consumed energy (in kilojoule), number of visited CHs, and number of deployed UAVs performances, as functions of the battery capacity, for different UAV trajectory planning approaches, namely Tabu search (Tabu), simulated annealing (SA), and guided local search (GLS). Performances of these methods are almost similar, with a preference for Tabu. Indeed, Tabu consumes less or equal energy to SA and GLS. Also, as the battery capacity increases (above 2250 mAh), the performance saturates, guaranteeing that data is collected from all CHs. Finally, the optimal battery capacity of 2500 mAh is provided by Tabu, where minimum energy (=167.9 kJ) and number of UAVs (=12) are achieved.

In Figs. 5–6, the same aforementioned performances are depicted as functions of the time deadline, for variations of Algo. 1, including "Algo. 1 (Short)" and "Algo. 1 (Range)", where in the latter, UAVs do not hover directly above CHs, but rather within a range of 150 m from the initial UAV locations defined in Algo. 1. For all approaches, as the deadline increases, the consumed energy reaches a peak then decreases. Below this peak, the deadline is very small such that it prevents collecting data from all CHs as the latter will be outdated, hence, a small number of UAVs is deployed, which consumes low energy due to short trajectories. In contrast, beyond the peak point, the deadline is long enough to deploy few UAVs that visit all CHs. The peak point corresponds to the critical deadline for which the maximum number of UAVs is deployed to collect data from the CHs. "Algo. 1 (Range)" presents the best energy consumption for deadlines above 90 sec, while visiting and deploying similar numbers of CHs and UAVs respectively, as Algo. 1. Indeed, "Algo. 1 (Range)" compensates for the prolonged data collecting time and energy (due to longer distances to CHs) by shorter flying time and energy. For deadlines below 90 sec, Algo. 1 performs either better or similarly to "Algo. 1 (Range)", in terms of energy and number of visited CHs, since hovering exactly above CHs improves the data transmission and thus respects the deadline. Finally, "Algo. 1 (Short)" presents the worst performances as it spends more energy to reach CHs at distant locations, or in contrast, abandon them due to their data becoming outdated.

VI. Conclusion

In this paper, we investigated the multi-UAV data collection problem in clustered IoT networks, where sensed data have time deadlines. Aiming to optimize the data collection

[3]This is valid assuming that all UAVs are of the same type, i.e., having the same mechanical and communication characteristics.

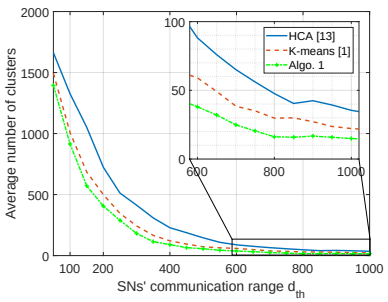


Fig. 1: Avg. no. of clusters vs. d_{th}.

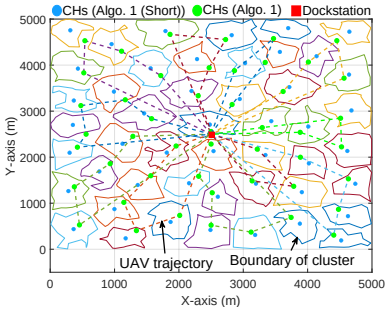


Fig. 2: Clustering and UAV trajectories.

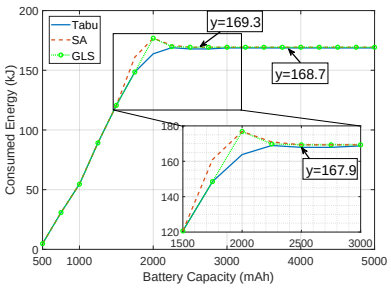


Fig. 3: Energy vs. battery capacity.

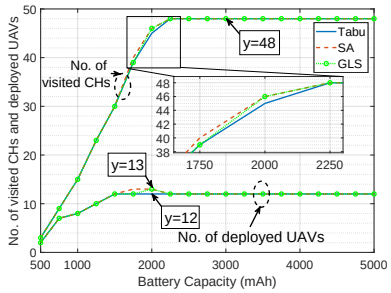


Fig. 4: No. of CHs and UAVs vs. battery capacity.

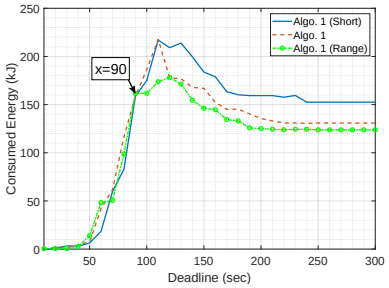


Fig. 5: Energy vs. deadline.

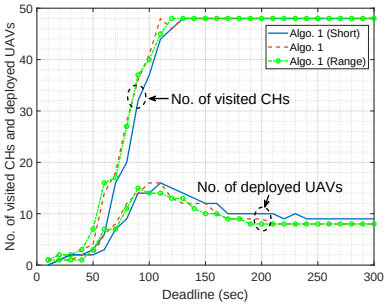


Fig. 6: No. of CHs and UAVs vs. deadline.

TABLE I: Simulation parameters

UAV altitude	H=100 m	Bandwidth	W=10 MHz
UAV speed	v_u=30 m/sec	Speed of light	V=3 $\times 10^8$ m/sec
Distance threshold	d_{th}=600 m	Mission time	T_f=4 min
Carrier frequency	f_c=2 GHz	Environment parameters	a=9.61 b=0.16
Time deadline	$T_{d,c}$=140 sec	Max SNs per cluster	F=120
Noise power	σ^2=–109 dBm	Packet size	S_p=1 Kbyte
TS length	δ=100 msec	Path loss	α=2.7
Battery capacity	$S_u(0)$=3,500 mAh	Number of packets	Q_c=50,000

deployment costs in terms of energy consumption, we propose a two-step solution. In the first step, the number and locations of CHs, which gather data from associated IoT sensors, are optimized using a customized K-means approach. Subsequently, an energy-efficient data collection framework which uses the minimal number of UAVs is presented; in this framework, the UAV trajectories are defined with respect to time deadlines of collected data and energy constraints. Simulation results show the efficiency of our customized K-means clustering compared to baseline approaches. Also, moving CHs closer to the dockstation provided a significant energy gain. On the other hand, it is shown that Tabu search achieves the best UAV trajectory design, compared to other methods. Finally, the impact of the battery capacity and time deadline is studied in terms of energy consumption, number of visited CHs, and number of deployed UAVs.

REFERENCES

[1] S. Alfattani, W. Jaafar, H. Yanikomeroglu, and A. Yongacoglu, "Multi-UAV data collection framework for wireless sensor networks," in *Proc. IEEE Glob. Comm. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[2] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu, "3-D placement of an unmanned aerial vehicle base station (UAV-BS) for energy-efficient maximal coverage," *IEEE Wireless Commun. Lett.*, vol. 64, no. 6, pp. 434–437, Aug. 2016.

[3] N. Cherif, W. Jaafar, H. Yanikomeroglu, and A. Yongacoglu, "On the optimal 3D placement of a UAV base station for maximal coverage of UAV users," in *Proc. IEEE Glob. Comm. Conf. (GLOBECOM)*, Dec. 2020.

[4] W. Jaafar, S. Naser, S. Muhaidat, P. C. Sofotasios, and H. Yanikomeroglu, "Multiple access in aerial networks: From orthogonal and non-orthogonal to rate-splitting," *ArXiv*, 2020. [Online]. Available: https://arxiv.org/abs/2005.13122

[5] M. Samir, S. Sharafeddine, C. M. Assi, T. M. Nguyen, and A. Ghrayeb, "UAV trajectory planning for data collection from time-constrained IoT devices," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 34–46, Jan. 2020.

[6] C. Zhan, Y. Zeng, and R. Zhang, "Energy-efficient data collection in UAV enabled wireless sensor network," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 328–331, Jun. 2018.

[7] A. T. Albu-Salih and S. A. H. Seno, "Energy-efficient data gathering framework-based clustering via multiple UAVs in deadline-based WSN applications," *IEEE Access*, vol. 6, pp. 72 275–72 286, Nov. 2018.

[8] A. Al-Hourani, K. Sithamparanathan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.

[9] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2329–2345, Mar. 2019.

[10] J. K. Lenstra and A. H. G. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, 1981.

[11] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. Ann. Hawaii Int. Conf. Syst. Sci.*, Jan. 2000.

[12] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient K-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. and Machine Intelli.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[13] F. Nielsen, *Ch. 8: Hierarchical Clustering - Introduction to HPC with MPI for Data Science*. Springer, 2016.

[14] D. Knuth. Capacitated vehicle routing problem, OR-tools. [Online]. Available: https://developers.google.com/optimization/routing/cvrp

Curiginal

Document Information

Analyzed document	GhdiriOussama_Report.pdf (D98460512)
Submitted	3/15/2021 11:13:00 PM
Submitted by	
Submitter email	jihene.benabderrazek@esprit.tn
Similarity	5%
Analysis address	jihene.benabderrazek.esprit@analyse.urkund.com

Sources included in the report

SA	**Aman Soni (BT16ECE014)_Final Year Project Report.pdf** Document Aman Soni (BT16ECE014)_Final Year Project Report.pdf (D76395797)	1
SA	**Using Reinforcement Learning for Dynamic Priority Assignment in Time Sensitive Net ...** Document Using Reinforcement Learning for Dynamic Priority Assignment in Time Sensitive Net ... (D95739316)	1
W	URL: https://www.researchgate.net/publication/343317699_AoI-Minimal_Trajectory_Planning ... Fetched: 3/15/2021 11:15:00 PM	2
W	URL: https://www.researchgate.net/publication/335873146_UAV_Trajectory_Planning_for_Dat ... Fetched: 3/15/2021 11:15:00 PM	1
W	URL: http://iot-analytics.com/top-10-iot-applications-in-2020/ Fetched: 3/15/2021 11:15:00 PM	1
W	URL: https://arxiv.org/pdf/1810.07862 Fetched: 12/11/2019 2:09:26 PM	1
SA	**elsarticle-RL.pdf** Document elsarticle-RL.pdf (D78755798)	6
W	URL: https://arxiv.org/pdf/1910.09281 Fetched: 1/5/2020 2:03:20 PM	5
W	URL: https://repository.tudelft.nl/islandora/object/uuid:4d02e51a-0187-404d-b465-7ae01f ... Fetched: 12/18/2020 11:39:54 AM	2
W	URL: https://etd.ohiolink.edu/apexprod/rws_etd/send_file/send%3Faccession%3Ducin1479816 ... Fetched: 1/14/2021 6:26:12 PM	1
W	URL: https://www.scss.tcd.ie/publications/theses/diss/2019/TCD-SCSS-DISSERTATION-2019-0 ... Fetched: 2/17/2021 4:16:45 PM	1
W	URL: https://www.scss.tcd.ie/publications/theses/diss/2018/TCD-SCSS-DISSERTATION-2018-0 ... Fetched: 1/3/2020 11:14:27 PM	4

ESPRIT SCHOOL OF ENGINEERING

www.esprit.tn - E-mail : contact@esprit.tn

Siége Social : 18 rue de l'Usine - Charguia II - 2035 - Tél. : +216 71 941 541 - Fax. : +216 71 941 889

Annexe : Z.I. Chotrana II - B.P. 160 - 2083 - Pôle Technologique - El Ghazala - Tél. : +216 70 685 685 - Fax. : +216 70 685 454