

Cooperative Data Muling Using a Team of Unmanned Aerial Vehicles

Emmanuel Tuyishimire (temmanuel@uwc.ac.za)

Thesis presented for Doctor of Philosophy

Department of Computer Science

Faculty of Natural Science



Supervisors:

Prof. Bigomokero Antoine Bagula (University of the Western Cape)

Prof. Nouleddine Boudriga (University of Carthage)

July 31, 2019

Abstract

Unmanned Aerial Vehicles (UAVs) have recently offered significant technological achievements. The advancement in related applications predicts an extended need for automated data muling by UAVs, to explore high risk places, ensure efficiency and reduce the cost of various products and services. Due to advances in technology, the actual UAVs are not as expensive as they once were. On the other hand, they are limited in their flight time especially if they have to use fuel. As a result, it has recently been proposed that they could be assisted by the ground static sensors which provide information of their surroundings. Then, the UAVs need only to provide actions depending on information received from the ground sensors. In addition, UAVs need to cooperate among themselves and work together with organised ground sensors to achieve an optimal coverage. The system to handle the cooperation of UAVs, together with the ground sensors, is still an interesting research topic which would benefit both rural and urban areas.

In this thesis, an efficient ground sensor network for optimal UAVs coverage is first proposed. This is done using a clustering scheme wherein, each cluster member transmits its sensor readings to its cluster head. A more efficient routing scheme for delivering readings to cluster head(s) for collection by UAVs is also proposed. Furthermore, airborne sensor deployment models are provided for efficient data collection from a unique sensor/target. The model proposed for this consists of a scheduling technique which manages the visitation of UAVs to target. Lastly, issues relating to the interplay between both types of sensor (airborne and ground/underground) networks are addressed by proposing the optimal UAVs task allocation models; which take caters for both the ground networking and aerial deployment.

Existing network and traffic engineering techniques were adopted in order to handle the internetworking of the ground sensors. UAVs deployment is addressed by adopting Operational Research techniques including dynamic assignment and scheduling models. The proposed models were validated by simulations, experiments and in some cases, formal methods used to formalise and prove the correctness of key properties.

Declaration

I declare that *Cooperative Data Muling Using a Team of Unmanned Aerial Vehicles* is my own work, that has not been submitted for any degree or examination in any other university.

I also declare that all the sources I have used or cited have been indicated and acknowledged by complete references.

July 31, 2019

Emmanuel Tuyishimire

Signed:



UNIVERSITY *of the*
WESTERN CAPE

Dedication

Gratien Gatabazi's family to which I belong.



UNIVERSITY *of the*
WESTERN CAPE

Acknowledgment

This thesis has only been possible because of the efforts of many people that have helped me through the years. I would first like to thank my supervisor Prof. Bigomokero Antoine Bagula who has been encouraging me during my studies. Without our frequent meetings, this thesis would not have been possible. Thank you Prof. Nouredine Boudriga for your invaluable co-supervision. Your technical comments and advice hugely helped me for this research. Thank you Dr. Slim Rekhis for kind reviews, comments and advice during this research.

My very grateful thanks go to the NRF through Coe-Mass bursary, Telecom funding facilitated by Dr. Mehrdad Ghaziasgar, and the UWC research office, who financially assisted me to complete this research. I would thank the whole UWC community especially the Computer Science department group for the various ways they have provided assistance for this research.

I would like to extend a very special thanks to my inspiring Mathematics teachers especially Augustin Ukobizaba who convinced me that I could do Mathematics at a very young age. Without their encouragement and coaching, I could not be where I am today. It is difficult to write the long list of all the people who contributed to the completion of this journey, though I do remember each and every one of them. I do remember those who spent their time praying for me, those who morally supported me during hard times, and those who used to help me by being my support team and would not allow me to say thanks.



Publications

Journal papers

- [1] **Emmanuel Tuyishimire**, Antoine Bagula, and Adiel Ismail. Clustered data muling in the internet of things in motion. *Sensors*, 19(3), 2019. ISSN 1424-8220. doi: 10.3390/s19030484.
- [2] Adiel Ismail, Bigomokero Antoine Bagula, and **Emmanuel Tuyishimire**. Internet-of-things in motion: a UAV coalition model for remote sensing in smart cities. *Sensors*, 18(7), 2018. ISSN 1424-8220. doi: 10.3390/s18072184.

Conference papers

- [1] Adiel Ismail, **Emmanuel Tuyishimire**, and Antoine Bagula. Generating dubins path for fixed wing uavs in search missions. In *Boudriga N., Alouini MS., Rekhis S., Sabir E., Pollin S. (eds) Ubiquitous Networking. UNet 2018. Lecture Notes in Computer Science*, volume 11277. Springer, Cham, 2018.
- [2] **Emmanuel Tuyishimire**, Antoine Bagula, and Adiel Ismail. Optimal clustering for efficient data muling in the internet-of-things in motion. In *Boudriga N., Alouini MS., Rekhis S., Sabir E., Pollin S. (eds) Ubiquitous Networking. UNet 2018. Lecture Notes in Computer Science*, volume 11277. Springer, Cham, 2018.
- [3] **Emmanuel Tuyishimire**, Antoine Bagula, Slim Rekhis, and Nouredine Boudriga. Cooperative data muling from ground sensors to base stations using uavs. In *Computers and Communications (ISCC), 2017 IEEE Symposium on*, pages 35–41. IEEE, 2017.
- [4] **Emmanuel Tuyishimire**, Adiel Ismail, Slim Rekhis, Antoine Bagula, and Nouredine Boudriga. Internet of things in motion: A cooperative data muling model under revisit constraints. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCOM/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pages 1123–1130. IEEE, 2016.
- [5] Antoine Bagula, **Emmanuel Tuyishimire**, Wadepoel Jason, Boudriga Nouredine, and Rekhis Slim. Internet-of-things in motion: A cooperative data muling model for public safety. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCOM/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pages 17–24. IEEE, 2016.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Deployment scenarios	2
1.3	Sensor network monitoring	3
1.4	Introduction to Z notation	5
1.5	Thesis Motivation and contributions	8
1.6	Declaration of publications	9
1.7	Thesis organisation	11
2	UAV-aware topology optimisation	12
2.1	Introduction	12
2.2	Problem Formulation	15
2.3	The Proposed Clustering Models	19
2.4	Issues and Relaxation	26
2.5	Results and Discussion	29
2.6	Conclusions	38
3	UAV-aware data routing	39
3.1	Introduction	39
3.2	Observables and problem definition	41
3.3	Proposed solution	46
3.4	Example	51
3.5	Experimental comparison of LIBA and LISPA	56
3.6	Conclusion	62
4	Cooperative Model for one Target Visitation	63
4.1	Introduction	63
4.2	The Cooperative Data Muling Model	65
4.3	Heuristic solutions	68
4.4	Collision recovery	71
4.5	Simulation and Analysis	73
4.6	Effect of the revisit deadline (r)	77



4.7	Delay comparison	78
4.8	Conclusion	79
5	Multiple Target Coverage	80
5.1	Introduction	80
5.2	The Cooperative Data Muling Model	81
5.3	Experimental results	88
5.4	Conclusion	104
6	Conclusion and future work	105
6.1	Conclusion	105
6.2	Future Work	105
	References	115

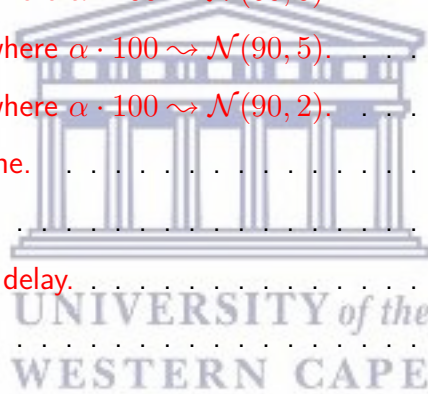


UNIVERSITY *of the*
WESTERN CAPE

List of Figures

1.1	Hybrid network partition as a combination of both the restricted (see the network with red links) and unrestricted network (see the network with black links).	3
1.2	class <i>Eve</i>	8
1.3	Contribution summary.	9
1.4	Chapter dependency.	11
2.1	Comparison between Distance-based Crowdedness Clustering (DCC) and deterministic k-means [1]. (a) Clustered nodes; (b) minimum energy per round. TEC, Total Energy Consumption.	14
2.2	Terrestrial, airborne, and hybrid networks. (a) Physical topology; (b) conceptualized topology.	16
2.3	Energy required versus the number of clusters: (a) $30m \times 30m$ network with 100 nodes ; (b) $60m \times 60m$ network with 100 nodes; (c) $30m \times 30m$ network with 200 nodes.	21
2.4	Beginning steps. (a) Step 0 : initial network; (b) Step 1 .	24
2.5	Processing steps. (a) Step 2 ; (b) Step 3 .	24
2.6	Last steps. (a) Step 4 ; (b) Step 5 .	25
2.7	Relaxation: energy inefficiency. (a) Non-relaxed clustering; (b) Distance-aware clustering.	26
2.8	Relaxation: orphan nodes. (a) Non-relaxed clustering; (b) relaxed clustering.	27
2.9	Case study. (a) GPS positions; (b) positions on the map.	30
2.10	Processed networks . (a) Communication network considered; (b) UAV path network considered.	30
2.11	Impact of parameters on performance. (a) Cost versus radius; (b) coverage cost versus the number of clusters.	31
2.12	Impact of UAV on clustering. (a) Radius-cost; (b) communication network.	32
2.13	Impact of the cluster-head selection policy.	33
2.14	Algorithms comparison on different topologies. (a) Case 1; (b) Case 2; (c) Case 3; (d) Case 4; (e) Case 5. UADC, UAV-Aware DCC.	34
2.15	Algorithms difference based on different topologies. (a) Case 1; (b) Case 2; (c) Case 3; (d) Case 4; (e) Case 5.	36
2.16	K-means algorithm. (a) Average disconnectedness: Cape Town network; (b) average disconnectedness: random network.	37
3.1	The system visualisation.	41
3.2	Routing with LIBA.	44
3.3	Node weights update issues.	45

3.4	data delay issue.	46
3.5	First iteration of <i>LISPA</i> .	53
3.6	Second iteration of <i>LISPA</i> .	54
3.7	Load balancing.	58
3.8	Delay evolution.	59
3.9	Delay evolution.	59
3.10	Delay evolution.	59
3.11	Highest cost variation.	60
3.12	Effect of the parameter α .	61
3.13	Centroid variation.	61
4.1	Visualisation of UAVs on action.	65
4.2	Collision visualisation.	71
4.3	Algorithms comparison for perfect system.	73
4.4	Comparison for the case where $\alpha \cdot 100 \rightsquigarrow \mathcal{N}(60, 5)$.	74
4.5	Comparison for the case where $\alpha \cdot 100 \rightsquigarrow \mathcal{N}(90, 5)$.	75
4.6	Comparison for the case where $\alpha \cdot 100 \rightsquigarrow \mathcal{N}(90, 2)$.	75
4.7	Ordered UAVs on a timeline.	76
4.8	UAVs arriving time.	77
4.9	Waiting time and average delay.	77
4.10	Load balancing.	79
5.1	Cooperative Data Muling.	81
5.2	Initial step.	87
5.3	The best neighbour choice.	88
5.4	Delivery.	88
5.5	Considered networks.	89
5.6	Paths generation when the UAVs have the same speeds.	89
5.7	Paths generation when the UAVs have the different speeds.	90
5.8	Paths generation when speeds distribution is changed.	91
5.9	Cost at same speed.	92
5.10	Cost at different UAVs' speeds.	93
5.11	Cost related to a different speed distribution.	93
5.12	Variation of the cost with respect to the speed.	94



5.13 Variation of the cost with respect to the overdue time (α).	95
5.14 Variation of the cost with respect to the delay (β).	95
5.15 Variation of the cost with respect to the data collection rate.	96
5.16 Paths details (waiting=30).	97
5.17 Paths details (waiting=60).	97
5.18 Paths details (waiting= ∞).	98
5.19 Number of unvisited nodes.	99
5.20 Visitation constrained by the waiting time.	100
5.21 Number of unvisited nodes.	101
5.22 Consecutive visitations.	102
5.23 Unconstrained visitations.	103



List of Tables

2.1	Parameters and their corresponding values.	20
2.2	Average disconnectedness: UAKM algorithm.	38
5.1	Data collection scenarios.	84
5.2	Data delivery when all drones have the same speed.	90
5.3	Data delivery when all drones have different speeds.	91
5.4	Data delivery when speed distribution changes.	91
5.5	Data delivery when UAVs may be delayed by no more than 30 min.	96
5.6	Data delivery when UAVs may be late for no more than 60 min.	97
5.7	Data delivery when UAVs may be late indefinitely.	98
5.8	Restricted data delivery.	101



1. Introduction

1.1 Motivation

In this section, the key motivating applications for conducting this research are highlighted.

1.1.1 Motivation. The use of UAVs started a long time ago [2] and was limited to military usage. Nowadays, the use of UAVs is justified in situations where the task to be performed is too dangerous, expensive or difficult to be performed by human beings [3], or, if not too difficult, can in any case be performed more cheaply and/or efficiently by UAVs. There are many such applications which are the subject of ongoing research and development.

Target search is a common application for the purposes of rescue [4, 5], monitoring [6, 7] or destruction [8]. Another popular application is that of area coverage or exploration for multiple purposes such as environment mapping [9], surveillance [10, 6, 11, 12], sensor deployment [13], acting as communications hubs for immobile wireless sensor networks [13, 14], smart city assistantships [15], or aero-biological sampling [16].

These and other research works mention applications such as weather forecasting, fire detection and observation (in both urban and rural environments), environmental clean-up, space exploration, traffic surveillance, logistics in warehouses and factories, agricultural monitoring and interior surveillance of buildings.

New and unforeseen applications also continue to surface [17]. There have been efforts in Israel and Australia to search for groundwater through the use of aerial drones. More recently, commercial applications using drones have emerged in niche areas where these vehicles have been found much more practical and more economically sound than traditional methods or approaches. These include the case where drones are being used commercially i) as marketing gimmicks [18], ii) to deliver pizza by the Dominos pizza company in the US [19], iii) to deliver flowers on Valentines Day [20], iv) to deliver urgently needed blood to hospitals in East Africa [21], and v) to deliver beer during a festival [22]. As another typical drones application, it has been reported that camera drones have been deployed to help combat the scourge of rhino-poaching on game farms in Africa [23], while the Australian defence force has started using these vehicles to monitor the coastline along its vast ocean borders [24].

With their recent acquisitions of aerospace companies (Ascenta by Facebook and Titan Aerospace by Google), Facebook and Google are investing in drones for supplying remote areas on the planet earth with broadband and Internet connectivity. The use of a team of drones instead of a single drone is possible when the mission to be performed can be broken down into basic tasks, i.e. the mission is inherently distributed in terms of space, time and functionality [25].

These applications along with the potential for drones to deliver life-saving medicines and supplies to isolated communities (rural and disaster zones) when overland access is not an option makes the multi drone task allocation an interesting research area that may benefit both urban and rural areas of the world [26].

1.1.2 Multiple UAVs usage. Multiple UAVs usage has been proven to be advantageous in many different ways [27, 28, 29, 30, 31, 26]. They may be classified according to one of three properties. (1) For **efficiency**, it is especially advantageous to use multiple UAVs when tasks are not order-dependant, because then multiple tasks can be performed simultaneously and the mission will take less time to complete. (2) On **robustness through redundancy**, the energy and computation required to perform the mission is distributed across team members, so if one member ceases to function, its role can be

assigned to another member. (3) Lastly for **flexibility**: the required functionality can be distributed across robots, although this advantage only becomes apparent when the mission is composed of diverse tasks: a team of robots with different specialisations, i.e. a heterogeneous team, can be engineered more cheaply and easily than a single robot capable of performing each kind of task.

1.2 Deployment scenarios

UAVs and ground sensors can be used in many deployment scenarios and tasked to achieve different goals in a multilayer network including airborne and terrestrial layers. The terrestrial layer consists of a static sensor network which collects local data to be forwarded to the patrolling UAVs. The UAVs form the airborne layer where a mobile network created by the UAVs can move to collect their readings [32, 33]. The two layers needs to be individually and collectively studied, alongside their interaction in order to investigate how optimal UAV coverage can be achieved.

1.2.1 Airborne deployment models. Two types of deployments are considered and are described as follows.

Single UAV deployment

UAV- sensor. This situation is when a single UAV is given a number of points to visit in a defined area that has been equipped with sensor devices, and the UAV is required to collect environmental variables at these specified points, or take pictures, if equipped with a camera sensor.

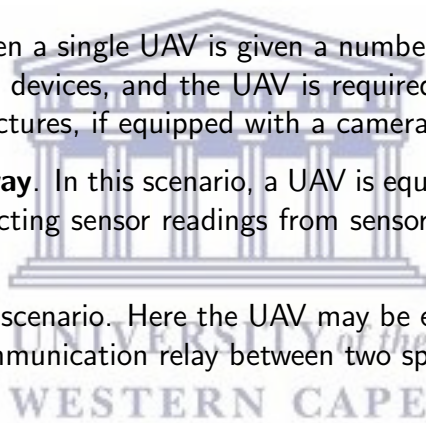
Another deployment is **UAV-gateway**. In this scenario, a UAV is equipped with a collection device, such as a sensor gateway, aimed at collecting sensor readings from sensors located at specified points in the defined area.

UAV-relay is another deployment scenario. Here the UAV may be equipped with a relaying device and play the role of an intermittent communication relay between two specified points in the area.

Multi-UAVs deployment

Deployment scenarios for multi drone task allocation may include **UAVs-sensor**, representing the case where a number of UAVs have to visit a number of points in an area which are equipped with sensor devices. The UAVs are required to collect environmental variables at the specified points or take pictures if equipped with a camera sensor. In contrast to the previous examples of a single UAV carrying out a simple search or visitation, these vehicles will need to be able to avoid collisions and restricted air zones due to regulations. Among other requirements, these UAVs will have to deal with the possibilities of danger, losses and failures. **UAVs-gateway** is another option, and consists of UAVs which are equipped with collection devices (sensors gateway) that can collect sensor readings from sensors located at the specified points within an area. These UAVs also need to be able to avoid collisions as well as be able to cope with previously mentioned situations. Finally, **UAVs-relay** is the situation where UAVs are equipped with relaying devices and play the roles of intermittent communication relays between pairs of specified points in the area.

1.2.2 Terrestrial deployment models. Three types of terrestrial networks are considered. They are described as follows.



Unconstrained network deployment

To explore a region, UAVs do not need to visit each and every sensor. It is better if they are assisted with a ground sensor network. The ground sensor network collects local readings and gathers together the data to be collected and interpreted by UAVs. Two scenarios are involved in this case. The first is a centralised deployment, where, a selected number of ground sensors act as the ground gateways, which are the ones that forward data to the UAVs. This would be beneficial in situations where a large sensor network needs to be monitored by a relatively small number of UAVs. The second scenario is a distributed deployment, where all the UAVs may have access to all the ground sensors in order to collect information from the ground sensor network. This is applicable in the case where the number of UAVs is sufficient to visit all sensors.

Constrained network deployment

In cases where there are obstacles, for illustrative purposes, given a UAV and three nodes x, y and z . The UAV may be unable to visit node x directly from node y without passing through node z . This might be caused by obstacles (such as houses, mountains etc.) in the region being explored. In such cases given a set of ground sensor positions allocated to a single UAV to cover, each UAV would have limited direct access to any other parts of the ground sensor network. Note that restrictions on path followed by the UAVs are based on obstacles.

Hybrid network deployment.

Note that both the unrestricted sensor network and the restricted sensor network depend on how the ground sensors communicate with each other. The hybrid network is the network where sensors can communicate with each other and can also be visited by UAVs subjected to permanent obstacles. Examples of the three networks are shown in Figure 1.1.

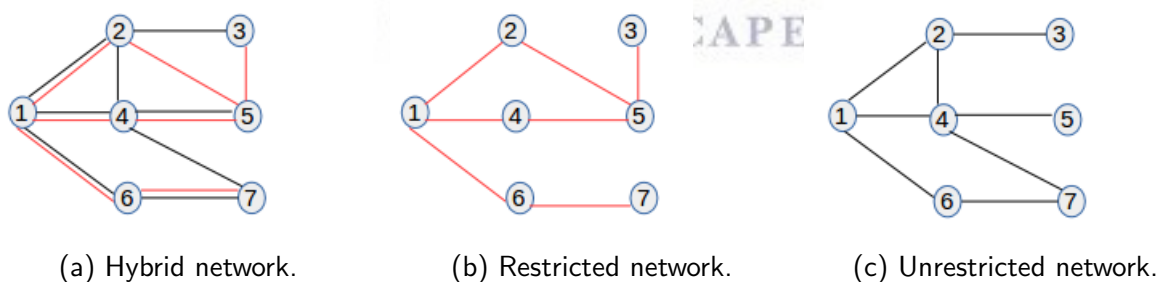


Figure 1.1: Hybrid network partition as a combination of both the restricted (see the network with red links) and unrestricted network (see the network with black links).

1.3 Sensor network monitoring

In this section, we discuss the network monitoring in two steps: the first step consists of discussing related network surveillance models and the last discusses the general problems which might be adopted to address the monitoring issues.

1.3.1 Network surveillance. As discussed in [34], UAVs are able to perform network surveillance by collecting information about intruders from Unattended Ground Sensors (UGS). The persistent surveillance in basic defence mission and pursuit is ensured by UGS revisit deadline handling. Here the revisit deadline is handled by minimizing the late and early visitations. This problem is mathematically formulated and proven to be intractable (NP-hard). The heuristic algorithm to coordinate the UAVs surveillance and pursuit is presented. Completeness and complexity of the algorithm is provided, and then, the effectiveness of the heuristic is illustrated by simulation.

It is assumed that the ground sensors cannot communicate with each other, but can communicate with UAVs, and also that the ground sensors are optimally placed before the mission, using the techniques discussed in [35] (alarm placement).

As stated in [34], No method treats a framework where the mobile agents fully rely on static sensors placed on an arbitrary road network to track and intercept intruders nor does the framework provide a path planning algorithm for the mobile agents. The paper [34] does not address the issues of obstacles and collision. Issues relating to obstacles have been addressed in [36, 37] by adopting the Particle Swarm algorithm. In [36], many robots aim to find a single target but the obstacle is unique and considered to be either a square or a circle. In [37], path finding in an unknown environment is handled. However, in neither paper is the collision issue addressed. The collision issue is handled in [36, 37] where the assumed environment is predictable. Here, UAVs are considered to be point masses whose trajectories are straight lines. UAVs are required to visit ground sensors in such a way as to reduce costs. This may be achievable if the selected nodes are able to collect data from other nodes and then forward these data to UAVs.

1.3.2 Network operational research.

- **Travelling salesman problem (TSP).**

The travelling salesman problem is detailed and discussed in a survey conducted during the last decade in [38]. The problem is formalised in different forms and it is proven to be NP-hard. The problem was able to be solved exactly for a 200-nodes network within a few minutes. A survey of the exact heuristics algorithms used to solve the problem is made. As indicated, the problem assumes the following:

- Hamiltonian graphs.
- The associated weight function satisfies the triangular inequality.
- An algorithm is presented in the paper as a heuristic which allows nodes to be visited more than once. In this case, the TSP is considered to be symmetric that is if c_{ij} is the cost to move from node i to j , then $c_{ij} = c_{ji}$.

It is possible to adopt heuristics to solve the Salesman problem which do not make these three assumptions. The TSP has been solved by successful use of a natural metaphor in [39], wherein behaviour of ants in colonies was used to address the obstacles issues. However, all ants are considered to have the same capacity which is an unnecessary assumption in the UAVs case.

- **Knapsack problem.**

The Knapsack problem, also called Rucksack problem, is a combinatorial optimisation problem defined as follows: given k items with their mass and values, and a bag of size W , determine the number of each item to include in the bag so that the total weight is less than or equal to a given limit W and the total value is as large as possible. The problem has been surveyed in [40], where heuristic and exact algorithms are provided. In some of the cases ([41, 42, 43], etc.) the number of items is either 0 or 1 and the corresponding problem is known as 0-1 Knapsack problem or binary Knapsack problem.

In case two, bags are to be used to collect items, the problem is called the 0-1 quadratic knapsack problem and it has been surveyed in [44] where exact and heuristic methods are discussed.

Given $k > 2$ bags, the problem is still an open challenge. In this case an overview of the problem is in [45]. Heuristics are presented and pseudo exact methods, with their limitations, are shown.

- **Generalized assignment problem.** The assignment problem is one of the fundamental combinatorial optimization problems in the branch of optimization or operations research in mathematics. It assumes a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. The requirement is to perform all tasks by assigning exactly one agent to each task and exactly one task to each agent in such a way that the total cost of the assignment is minimized. The assignment problem can take various forms some of which are known to be NP-hard. These include the Quadratic Assignment Problem [46] where, the cost related to the interaction of agents is considered; the Dynamic Assignment Problem [47], where the assignment cost for each agent may change with time due to the resources and agents dynamical availability over time. This is why the generalized assignment problem take care of all these cases and hence it is NP-hard. This problem has been exactly and approximately solved using various algorithms, which are surveyed in [48].
- **Vehicle routing problem.** Given a link weighted network, the problem consists of finding the cheapest route from a single depot while visiting every node only once and using one or more vehicles. Heuristic and exact algorithms are presented in [49]. The assumption here is that all the used vehicles are identical. The heuristics presented are for big networks, and it is mentioned that the better ones still need attention. For this optimisation problem, the main constraints are put into the following groups:
 - Time to visit each city
 - Time to visit all cities in one route
 - Capacity of every city
 - The number of cities at every route
 - Precedence of visiting the cities

Note that when dealing with one vehicle, the problem could be transformed into the TSP 1.3.2.

1.4 Introduction to Z notation

The Z notation [50] was developed, largely at Oxford in the early to mid 1980's to clarify the description of complex discrete transition systems (the analogue case is dealt with satisfactorily by differential equations). It views a transitions system, like $AODV$, as a state-based system with operations on it and this provides techniques for the structured description of state and for the structured description of the operations on state (with input and output). Here we give a summary of the notation, referring to [50] for details.

The state S of a discrete system consists of several typed observables, vector $v : V$, subject to some invariants P (where P is a predicate with free variables v). This is expressed by schema:

S
$v : V$
P

For example a system *Even* whose state consists of just an even integer n is written

<i>Even</i>
$n : \mathbb{Z}$
$\exists m : \mathbb{Z} \bullet n = 2m$

In writing P , conjuncts are written on separate lines, with \wedge omitted.

Schemas may be nested by using a schema as a type V . An observable $v : V$ regarded as input is written $v? : V$ and one regarded as output is written as $v! : V$ (notation inherited from process algebra). State-before is written s and state-after written s' for each observable s .

Each system operation Op takes a state $s : S$ before and an input $in? : In$ and returns a state $s' : S$ after and an output $out! : Out$, subject to the invariant property $Q(s, s', in?, out!)$:

Op
$s, s' : S$
$in? : In$
$out! : Out$
Q



For convenience ΔS is defined to be

ΔS
S
S'

Note that inclusion of ΔS brings into scope the observables s and s' of type S , for use in predicate Q . Both s and s' are automatically constrained by the predicate P in S , which does not need to be repeated. In Z , a variable not explicitly constrained is assumed to take an arbitrary value (of its type). We therefore find that it is convenient to use the variation, due to object Z [51], $\Delta(vs)$ where state variables not in the vector vs of variables remain unchanged. Thus with the type *Even* for a pair of distinct even numbers,

$Evens$
$a, b : Even$
$a \neq b$

we have

$\Delta Evens$	=	$\Delta Even$
$Evens$		$a, a', b, b' : Even$
$Evens'$		$a \neq b$ $a' \neq b'$

whilst

$Evens(a)$
$\Delta Even$
$b = b'$

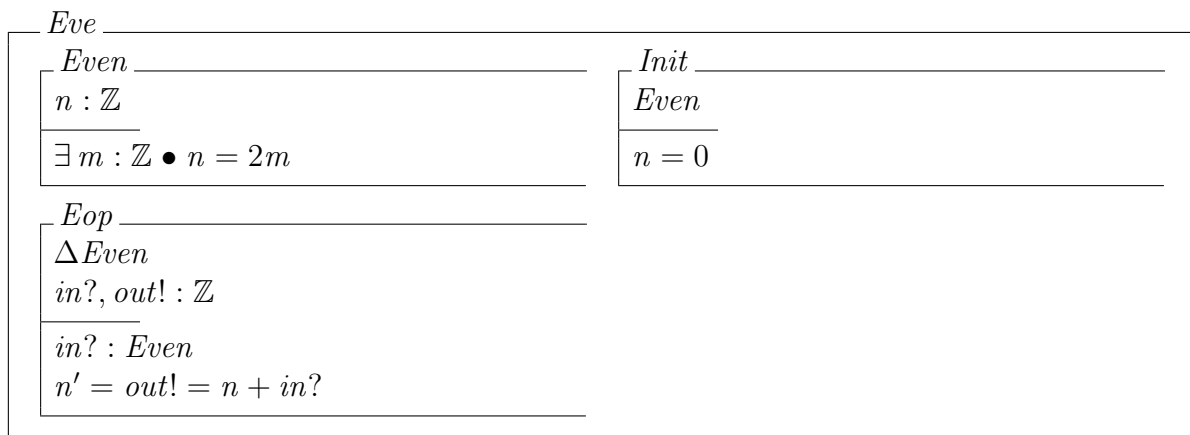
Projection notation is inherited from that for records in programming. If $x : Even$ then $x.a$ and $x.b$ denote the two components of x . Note that no ordering implied between the variables a and b of $Evens$. For example an operation which inputs an even integer, adds it to the state and outputs the result, is written

Eop
$\Delta Even$
$in?, out! : \mathbb{Z}$
$in? : Even$
$n' = out! = n + in?$

State is initialised by further constraining state to satisfy the initialization property; for example.

$Init$
$Even$
$n = 0$

Finally all ingredients are combined to produce a class consisting of state, initial state and operations. For example functions examples appear in the remainder of the report.

Figure 1.2: class *Eve*

1.5 Thesis Motivation and contributions

1.5.1 Motivation. Cooperating UAVs is a special case of using a multi-robot systems to perform a specific task, which is generally difficult to solve, especially as the number of degrees of freedom increases. In a typical UAV application, the number of used vehicles are to ensure the optimal coverage, subjected to different constraints, including: limited time, collision avoidance, limited speed and maximum acceleration. It has recently been shown in [34] that it is important to assist UAVs by using ground sensors to perform some of the tasks for the UAVs. Furthermore, optimal placement of ground sensors has been introduced in [35]. The system built by the two models ([34, 35]) was for surveillance and pursuit of a single intruder entering the network. The placement of sensors depended on maximizing the chances of capturing an intruder and sending information to UAVs regularly visiting these sensors. This recent system is still accompanied by assumptions, such as communication based ones, and requires improvement by considering more features, which include scenarios where the ground sensor network works together with the UAVs assignment, to achieve a more efficient coverage. Furthermore, optimal communication and exploration models for the ground sensor network need to be in place and customized. This thesis aims to build on the above mentioned system, and propose model for a such system for more efficient data transport from a ground sensor network to a point where the data could be processed further. In this thesis, communication among sensors is assumed guaranteed in order to produce data collection, transport and delivery with improved efficiency.

1.5.2 Contributions. The contributions of this thesis are summarised in Figure 1.3.

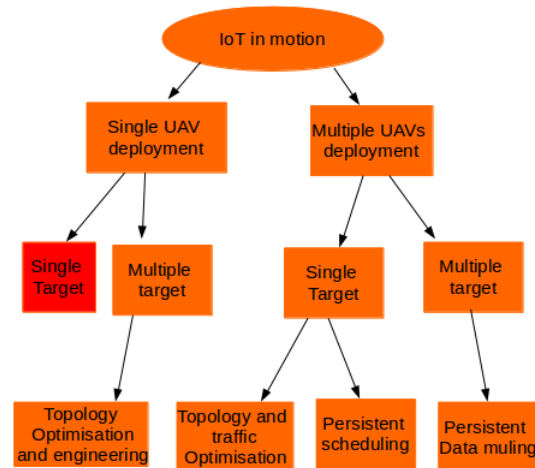


Figure 1.3: Contribution summary.

1. **Topology optimisation and engineering.** The first case to consider is a single UAV- Multi-sensor deployment, where one UAV has to visit a number of locations (of sensors). The locations are connected in two ways: first, with respect to the possible paths the UAV may take to visit sensors and secondly, with respect to the way the sensors can communicate. Well known classic problem/algorithms ([52, 53, 54]) have been used extensively to solve various scenarios of this type of problem. For all these scenarios, every location needs to be visited. However, in this work, a single UAV is constrained and cannot visit each and every sensor. Thus the key sensors to be visited by the UAV must be selected so as to minimise the data muling by UAVs. The selected key sensors need to collect local sensor readings and relay this information to the UAV as soon as it arrives. This is why, for this kind of deployment, it is necessary to use a clustering scheme for efficient data collection and transport.
2. **Topology and traffic optimisation.** We further propose a central routing model where local sensors need to send data to the sink which is assumed to be visited by many UAVs, coming from different directions. This is done in two steps: (i) the optimal sink/centroid which ensures efficient data gathering, transport and delivery by UAVs is determined; (ii) this is complemented by proposing an efficient routing algorithm, which is formalised, verified and analysed.
3. **Persistent scheduling.** Here, a many-UAVs-one-sensor deployment is considered which consists of many UAVs visiting a single location/sensor. This persistent visitation problem is addressed by proposing an efficient persistent scheduling scheme which is formalised, and analysed.
4. **Persistent data muling.** A multi-sensors-multi-UAVs deployment is considered. Here, many UAVs visiting various connected sensors and the UAV path planning is addressed. In fact, each sensor to be visited is assumed to be a local gateway, where local information is gathered. This model is applicable in a distributed deployment system, where many UAVs may visit all sensor nodes.

1.6 Declaration of publications

Some models and figures in the thesis have appeared in the following papers, which have been published.

Related publications

Journal papers

- [1] **Emmanuel Tuyishimire**, Antoine Bagula, and Adiel Ismail. Clustered data muling in the internet of things in motion. *Sensors*, 19(3), 2019. ISSN 1424-8220. doi: 10.3390/s19030484.
- [2] Adiel Ismail, Bigomokero Antoine Bagula, and **Emmanuel Tuyishimire**. Internet-of-things in motion: a UAV coalition model for remote sensing in smart cities. *Sensors*, 18(7), 2018. ISSN 1424-8220. doi: 10.3390/s18072184.

Conference papers

- [1] Adiel Ismail, **Emmanuel Tuyishimire**, and Antoine Bagula. Generating dubins path for fixed wing uavs in search missions. In *Boudriga N., Alouini MS., Rekhis S., Sabir E., Pollin S. (eds) Ubiquitous Networking. UNet 2018. Lecture Notes in Computer Science*, volume 11277. Springer, Cham, 2018.
- [2] **Emmanuel Tuyishimire**, Antoine Bagula, and Adiel Ismail. Optimal clustering for efficient data muling in the internet-of-things in motion. In *Boudriga N., Alouini MS., Rekhis S., Sabir E., Pollin S. (eds) Ubiquitous Networking. UNet 2018. Lecture Notes in Computer Science*, volume 11277. Springer, Cham, 2018.
- [3] **Emmanuel Tuyishimire**, Antoine Bagula, Slim Rekhis, and Nouredine Boudriga. Cooperative data muling from ground sensors to base stations using uavs. In *Computers and Communications (ISCC), 2017 IEEE Symposium on*, pages 35–41. IEEE, 2017.
- [4] **Emmanuel Tuyishimire**, Adiel Ismail, Slim Rekhis, Antoine Bagula, and Nouredine Boudriga. Internet of things in motion: A cooperative data muling model under revisit constraints. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCOM/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pages 1123–1130. IEEE, 2016.
- [5] Antoine Bagula, **Emmanuel Tuyishimire**, Wadepoel Jason, Boudriga Nouredine, and Rekhis Slim. Internet-of-things in motion: A cooperative data muling model for public safety. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCOM/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pages 17–24. IEEE, 2016.

1.7 Thesis organisation

Figure 1.4 shows the dependency of chapters in this thesis. Note: Chapter x is abbreviated to Chap x for $x=1$ to 6.

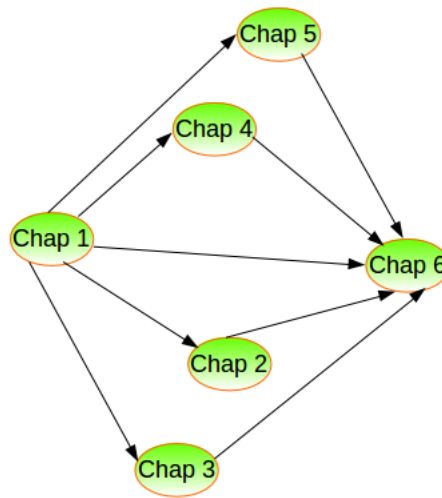


Figure 1.4: Chapter dependency.

As shown by Figure 1.4, the arrows pointing to a node n coming from the prerequisite chapter to understand the content of one shown by the node n . The tail of each arrow indicates a prerequisite chapter.

The above chapters are organised as follows.

After introducing the thesis in Chapter 1, a clustering scheme is proposed in Chapter 2. The UAV-aware data routing model is proposed in Chapter 3 and a scheduling model to visit one target is proposed in Chapter 4.

The multi-targets-multi-UAVs visitation is proposed in Chapter 5 and the thesis is concluded in Chapter 6.

2. UAV-aware topology optimisation

This chapter considers a case where an Unmanned Aerial Vehicle (UAV) is used to monitor an area. The UAV is assisted by a Sensor Network (SN) which is deployed in the area. The region to be monitored has a reasonable size and hence may contain many sensors for efficient and accurate data collection. In this case, it would be expensive for one UAV to visit each and every sensor. It is therefore important to partition the ground network into an optimum number of clusters, so that the UAV only has to visit cluster heads (i.e. fewer sensors). The collected data would be sent to cluster heads to be forwarded to the UAV, upon its arrival. In this chapter a clustering scheme is proposed, not only to optimise the WSN communication energy, but also the energy used by the UAV while covering the area. The optimal number of clusters in a dense and uniformly distributed sensor network is calculated, to support the k-means clustering. Furthermore, for general networks, an efficient clustering model that recognises nodes connectivity is proposed and analysed through simulations.

2.1 Introduction

The use of Unmanned Aerial Vehicles (UAVs) continues to be not only one of the most efficient approaches, but also less expensive and risky ones, for various exploratory problems. These problems include rescuing, data delivery/collection, surveillance, and many more. In the case of city surveillance, it has been found efficient to assist UAVs with a Sensor Network (SN) comprising static ground sensors, which collect local information and deliver them to the UAVs visiting them [34]. In case more detailed information is to be captured, large-scale and complex SNs are usually deployed in the zone of interest. In this case, the use of UAVs continues to be one of the most efficient ways to handle the mentioned situations. However, UAVs' flights are generally constrained by their limited flight time, fuel and energy usage when powered by battery. Therefore, the UAV exploration of targeted environments necessitates the optimization of energy usage to ensure the scalability and resilience of the data capturing. This is why it is generally important to minimize the UAVs' moves, yet collect maximal information by having the UAVs visit only an optimal number of selected ground sensors serving as ground gateways. Each ground gateway receives information collected from other ground sensor nodes for collection and data muling by a UAV upon its visit. It is then important to assign to each gateway an optimal team of sensor nodes providing the sensed data. Here, we refer to the teams of sensor nodes as cluster members, while their gateways are referred to as cluster heads, and the corresponding partition of the sensor network is called clustering.

2.1.1 Related Work. Cluster-based sensor networking has been a subject of high interest in the literature. In [55], the physical-access control cross-layer analytical approach for determining the optimum number of clusters has been proposed. The proposed model minimizes the communication-energy consumption in a highly-dense sensor network. In [56], the Euclidean distance (communication range and the area on which the network is deployed) from nodes to a cluster head was considered in order to design clusters with the objective of minimizing the energy required for efficient communication. In the latter paper, the energy usage is minimized with the increase of the number of clusters. A connectivity-based k-hop to the cluster head was proposed as a clustering technique in [57]. In the work, it was shown that the efficiency of messages transmissions from the cluster heads to the sink of the underlying network is reduced with the number of clusters. This raises the issue of finding the optimal number of clusters in a network (note that it exists). An optimal, temporal clustering algorithm was proposed in [58], as an adaptive model for a wireless micro-sensor network, to ensure efficient utilization of its energy. In [1], the optimal number of cluster heads and their locations were analytically computed for efficient wireless sensor network communication. The main goal of the paper was to ensure optimal data transfer in the

network by adopting the cluster head selection method in [59], which is based on the calculated probability of a node to be a cluster head. Simulations in [1] showed a better performing clustering, compared to the k -means algorithm-based [60] schemes, including those presented in [61, 62, 63, 64, 65].

The k -means is a clustering algorithm aiming to partition nodes into Voronoi cells (see [66] for example). Given the number k of centroids (cluster heads), the algorithm consists of the following steps.

1. Initialization: this is done by randomly selecting k nodes to be cluster heads.
2. Assignment step: this step consists of assigning cluster members to cluster heads, based on the least Euclidean distance between the node and cluster heads.
3. Update: for each cluster, a centroid (most central node) is computed, and if it is different from the current cluster head, it replaces it.
4. Iteration: this step consists of alternating Steps 2 and 3 until no more updates are possible.

The clustering problem being an NP-hard problem, the k -means is its heuristic solution, whose limitations include: (i) local convergence, (ii) the choice of the number k of cluster heads influencing the optimality of the clustering, (iii) the initialization step impacting the running time of the algorithm, and (iv) Euclidean distance used as the utility function for clustering. To address issues related to the above four properties, different versions of the algorithm have been proposed, respectively a globally-converging clustering [67], an optimal number of clusters for image segmentation [68, 69], a better initialized k -means [70] algorithm, and the multi-norm clustering [71]. However, to the best of our knowledge, there is no k -means algorithm that has been proposed to ensure the connectivity of all cluster members to corresponding cluster heads in order to avoid orphan/isolated nodes in a sensor network. Two versions of the k -means algorithm were considered in [1]: (i) the deterministic k -means algorithm, which is built around the same principles as the classical k -means algorithm, and (ii) the adaptive k -means algorithm, which uses the classical k -means algorithm iteratively to cluster n sensor nodes for n times by varying the parameter k from 1 to n and selecting the clustering result with the minimum energy cost. The Distance-based Crowdedness Clustering (DCC) was also proposed in [1] as a greedy algorithm, which, for a given general network, outputs the corresponding clustering by using node degrees as a way of selecting the best cluster head (one of highest degree) and building corresponding clusters. In DCC, the length of the underlying network's links is used to select the clustering radius (the length of one of the links), and every neighbor of the selected cluster head at a distance less than the radius is added to the cluster. This process continues to all remaining nodes, until each node belongs to a cluster. After performing clustering once, the corresponding cost (a function of the radius) is computed, and for all possible values of the radius, a clustering corresponding to the least cost is chosen to be the output of the algorithm.

Figure 2.1a shows a comparison of DCC and the adaptive k -means algorithms. In the figure, the solid lines represent clusters with the DCC, while the dotted lines represent the adaptive k -means clustering for 100 nodes, on a 200 m \times 200 m area. The figure reveals that DCC outperforms k -means in terms of cluster-based node density. Similarly, Figure 2.1b shows that the DCC outperforms the k -means in terms of Total Energy Consumption (TEC) efficiency for 10 consecutive runs of both algorithms.

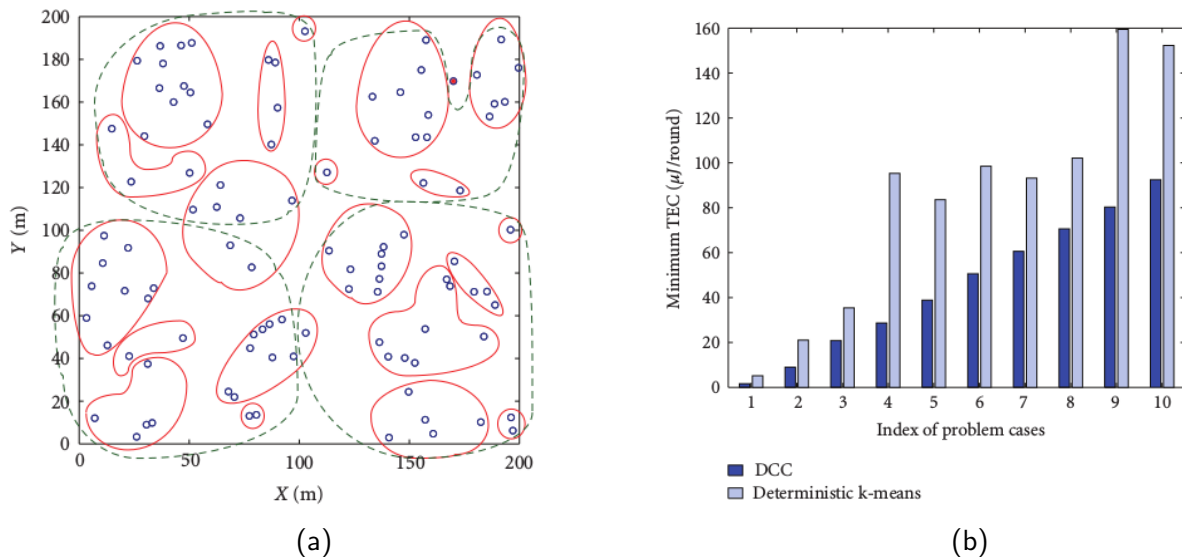


Figure 2.1: Comparison between Distance-based Crowdedness Clustering (DCC) and deterministic k-means [1]. (a) Clustered nodes; (b) minimum energy per round. TEC, Total Energy Consumption.

While the models and algorithms presented above focused the optimization process on a ground-based/terrestrial sensor network, the works in [72, 73, 74, 75, 76] are among those that have addressed UAVs' related clustering. In [72], UAVs were presented as moving agents, which were clustered by using their mobility attributes to predict their motion, hence leading to clusters' predictions. In [73], the UAVs also were clustered with the goal of computing the optimal route discovery. In [74], a clustering scheme was proposed to provide Internet connectivity, using a mobile sink (a UAV). In the paper, the clustering was performed based on the distance separating potential cluster heads and other ground sensor nodes, and also the proximity of the UAV, and a UAV's move was predicted in order to determine its corresponding cluster. To the best of our knowledge, this was one of the first clustering schemes considering different positions of a UAV (path). However, the paper did not consider the energy spent by the UAV while moving from one position to another, which is a requirement to allow the UAV to aggregate data as much as possible prior to being recharged. The works in [75, 76] considered a multi-layer model with a team of UAVs playing the role of an airborne gateway network for a terrestrial sensor network. While [75] proposed an initial model showing through simulation how the multi-layer network can be designed, the model in [76] was based on MIMO clusters to increase the terrestrial sensor network lifetime by avoiding disconnections that can lead to orphan/isolated sensors or groups of sensors that are unable to deliver their data.

2.1.2 Motivation and Contribution. As discussed in Section 2.1.1, the k-means algorithm (see [60] for example) and its variants are some of the most popular clustering models. This algorithm aims to minimize the sum of distances (standard deviations) between k cluster heads and their cluster mates. The deterministic k-means algorithm finds the best number k , and a clustering cost function may be used to evaluate the cost corresponding to each value of k ranging from one to the number of observations. Alternatively, mathematical methods using calculus are used to compute the number k . When the connection (affinity) between cluster members is one of the requirements, this algorithm is outperformed in terms of TEC, even in dense networks (see [1]). Note that for the k-means algorithm, nodes are grouped based on their statistical characteristics. However, statistical approaches alone could be less efficient in case the relationship of observations matters. The affinity of data points/nodes has been addressed in [77, 78, 79], but could not guarantee a perfect assignment of the node to the correct cluster head (the node to which all cluster members are connected). This is why the DCC algorithm proposed in [1] could outperform the adaptive k-means algorithm.

This chapter extends [80] to revisit the problem of clustering as a way of optimizing hybrid terrestrial/airborne sensor networks by proposing a novel clustering model that combines efficient sensor network communication and efficient cluster heads' visitation by a UAV. The clustering problem for a hybrid network (UAV routes and the communication-based SN) is firstly proposed. Thereafter, the optimal number of clusters is rigorously computed for uniform and dense network distribution settings. A heuristic clustering is then proposed for general networks; and its extension to cater for the sensor nodes' isolation is supported through relaxation techniques. Our work is closely related to DCC in [1], but differs by proposing a clustering scheme that (i) takes care of the relationship of nodes while DCC does not and (ii) considers a multi-layer approach that caters for the efficient cluster heads' visitation by a UAV. While different clustering schemes and algorithms have been proposed in the literature, they have either focused the optimization process on a single layer (UAV layer or terrestrial layer) [55, 56, 1, 60, 61, 62, 63, 64, 65] or consisted of non-optimization techniques that show how UAVs can be used as mobile sensor networks [34] for different purposes including city surveillance. Our model is based on an optimization process that considers both layers of a hybrid sensor network. Furthermore, the presence of orphan nodes (which could be either cluster heads or normal nodes) may lead to (i) a dislocated network with part of the data produced by the orphan nodes not reaching the network gateway and (ii) an energy-inefficient hybrid network with a UAV's energy being wasted to visit an orphan cluster head that does not have data to be collected. While all previous works have discounted the issue of orphan nodes, the clustering solution presented in this chapter addresses this issue by the proposed mitigation processes to reduce the number of orphan nodes.

The rest of the chapter is organized as follows. The problem is mathematically formulated in Section 2.2, and the proposed algorithmic solution is described in Section 2.3. To adopt a special case, the proposed algorithm is relaxed in Section 2.4, and the performance of the proposed model is discussed in Section 2.5, whereas in Section 2.6, the chapter is concluded.

2.2 Problem Formulation

In this section, the clustering problem is formalized as an energy optimization problem, under network-related constraints. The focus lies on an energy-efficient design where a single UAV located at a specific base station is used to collect sensor data from a number of collection points. The network \mathcal{H} can be considered as a hybrid network $\mathcal{H}(\mathcal{H}_g, \mathcal{H}_a)$ combining the terrestrial sensor sub-network \mathcal{H}_g and the airborne muling sub-network \mathcal{H}_a consisting of all possible UAVs' paths. Note that while having the same number of nodes, the \mathcal{H}_a network might differ from \mathcal{H}_g as it is based on potential UAV path restrictions related to obstacles and Distance-based Crowdedness Clustering (DCC) different environmental limitations. This is illustrated by Figure 2.2.

Figure 2.2 reveals that while the two network configurations in Figure 2.2a (aerial and ground networks) have the same sets of nodes, they may have different sets of links and hence different routing paths. Therefore, they may result in different energy consumption patterns ($\mathcal{E}_g \neq \mathcal{E}_a$). This raises the issue of energy consumption in a hybrid network (Figure 2.2b) and the need for an optimization model that combines the energy consumed by both networks $\mathcal{E}_h = f(\mathcal{E}_g, \mathcal{E}_a)$.

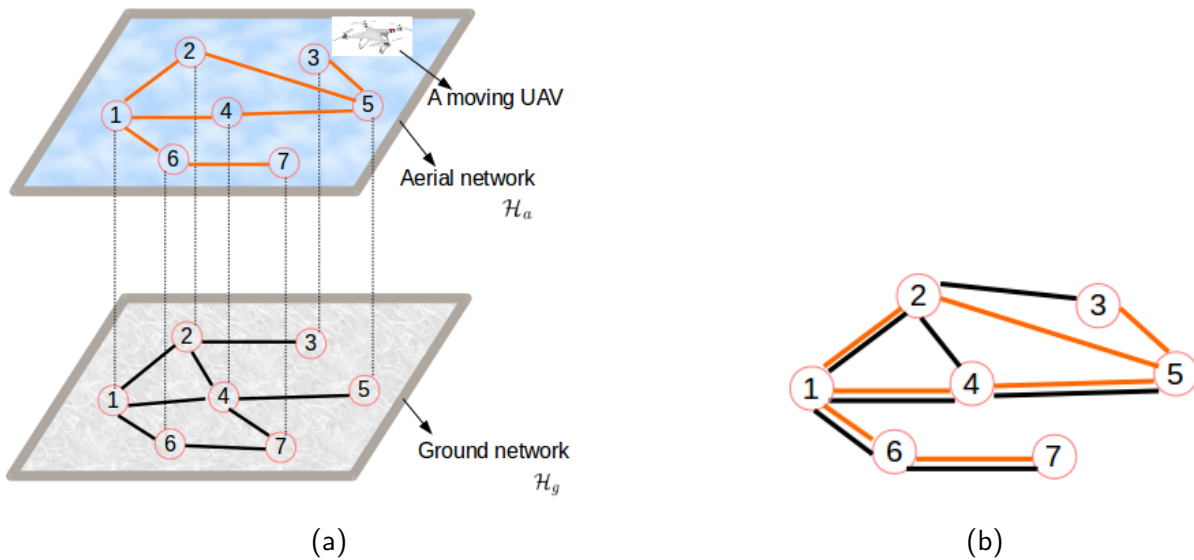


Figure 2.2: Terrestrial, airborne, and hybrid networks. (a) Physical topology; (b) conceptualized topology.

2.2.1 The Energy Models. As suggested earlier, this chapter considers an energy-efficient model where the energy consumption is described below.

$$E_g = E_t + E_r \quad (2.2.1)$$

$$E_a = \beta E_c + \gamma E_u, \quad (2.2.2)$$

where the constants β and γ are proportionality constants showing the weighting of E_c and E_u , respectively, and the energy components E_r , E_t , E_c , and E_u are defined below.

- Energy for sensor-data reception (E_r): This is the energy spent by cluster heads due to its topological and environmental properties, the physical/electronic properties of the receiving node, and the nature of messages to be received. We assume that all possible cluster heads are in the same and good condition; hence, they require the same quantity of energy to receive a message. It is assumed that nodes communicate directly with their corresponding cluster head, and in the case a multi-hop communication is applicable, the least interference beaconing protocol (see [81]) is used to find sensor communication route.
- Energy for data transmission among sensors (E_t): This is the total energy required to move the captured data from each cluster node to its corresponding cluster head. This form of energy is directly proportional to the distance separating the two communicating sensors. We assume one-hop inter-cluster communication, and hence, the considered distance is the Euclidean length of links. All nodes of the network are assumed to require the same quantity of energy for message transmissions.
- Energy for UAV data transport (E_u): This refers to the expected energy required for a UAV to visit cluster heads. This energy depends on the number of cluster heads in the H_g network and the distance between these nodes (the expected link length).
- Energy for UAV data collection (E_c): This is the energy spent by the UAV to collect data from the sensor nodes (cluster heads).

From Equation (2.2.1), the overall energy E_h for data dissemination in the terrestrial ground-based sensor network to cluster heads and data muling by the UAV can be expressed by the weighted sum of energy consumption in both ground and airborne networks as expressed by:

$$E_h = \alpha E_g + \beta E_u + \gamma E_c, \quad (2.2.3)$$

where, α represents the weighting of E_g

The Terrestrial Network Energy Consumption: E_g

Let $L \times L$ units of area be the area of the field where sensors are distributed. It follows that one cluster's area is $\sqrt{L^2/k} \times \sqrt{L^2/k}$, based on the Voronoi diagram (see [82]).

It has been shown in [1] that the total energy \mathcal{E} for data gathering in a uniformly-distributed network of type H_g is expressed as follows.

$$\mathcal{E} = (2n - 2k + a \cdot k)E_e + nE_p + (n - k)e_f \frac{L^2}{3k} + a \cdot k \cdot e_m \frac{4L^4}{9}, \quad (2.2.4)$$

where a (with $0 < a \leq 1$) denotes the data compression ratio: an input of k bits results in an output of $a \cdot k$ bits after compression; E_e denotes the energy for driving the electronics; E_p is the energy for data processing; n the number of all sensors in the field; and the constants e_f and e_m represent the coefficient corresponding to the effects of the clusters intra-distances and inter-distances, respectively.

The considered case in this chapter assumes that there is no inter-cluster communication, and thus,

$$e_m = 0.$$

This is why the gathering energy E_g for the uniformly-distributed network of type H_g is computed as follows.

$$E_g = (2n - 2k + a \cdot k)E_e + nE_p + (n - k)e_f \frac{L^2}{3k}, \quad (2.2.5)$$

On the other hand, for the generally-distributed network, the energy may be computed as follows. Let \mathcal{C} be a set of clusters; c_i represents the node i of cluster c , and c^h denotes the cluster head of cluster c .

$$E_g = (2n - 2k + a \cdot k)E_e + nE_p + \sum_{c \in \mathcal{C}} \sum_{i \in c} d(c_i, c^h), \quad (2.2.6)$$

where the function $d(c_i, c^h)$ represents the Euclidean distance between node i and the cluster head in the cluster c .

The Data Collection Energy Consumption: E_c

This is the total energy for data collection from cluster heads by a UAV. Let $1, 2, \dots, k$ be the indices corresponding to k cluster heads. If E_i is the energy required by the UAV to receive data from the cluster head i (with $1 \leq i \leq k$) and e_i is the energy required by the cluster head to forward the gathered data to the UAV, then the total energy E_c for data collection is expressed as follows.

$$E_c = \sum_{i=1}^k (E_i + e_i) = \frac{k}{k} \sum_{i=1}^k (E_i + e_i) \quad (2.2.7)$$

Hence,

$$E_c = k(\bar{E} + \bar{e}), \quad (2.2.8)$$

where \bar{E} and \bar{e} are the expected value of the energy required to receive and forward data, respectively.

Energy for UAV Transportation: E_u

The transportation energy E_t depends on the length of the used path, which gets longer as the number of clusters increases. We assume that the UAV moves from one node to another, using Dijkstra's algorithm [83] on the network of type \mathcal{H}_a . This will enable us to evaluate the goodness of a node to be in a particular cluster or even to be a cluster head.

Since the UAV-transportation energy is directly proportional to the length of the path used, it is also directly proportional to the number of cluster heads and the average distance from one cluster head to another and hence the distance D to travel from one node to another. Therefore, E_t is computed as follows,

$$E_t = b \cdot k \cdot D. \quad (2.2.9)$$

where b is the proportionality constant and $D = E(E_j(d))$ is the expected value of the average length of shortest paths d from each sensor node j to others, where $E_j(d)$ expresses the expected Dijkstra's shortest distance d from the node j to any node in the underlying network (here, it is \mathcal{H}_a).

Considering a network whose number of nodes is n , let $A_{n \times n} = \{d_{ij}\}$ be the matrix where each entry d_{ij} corresponds to the shortest distance from node i to node j based on Dijkstra's algorithm. Here, $d_{ii} = 0 \forall i$ because d_{ii} represents the distance from node i to itself. The index D may be calculated as follows.

$$D = \frac{1}{n-1} \sum_{j=1}^n \left(\frac{1}{n-1} \sum_{i=1}^n d_{ij} \right) = \frac{1}{(n-1)^2} \sum_{j=1}^n \sum_{i=1}^n d_{ij} \quad (2.2.10)$$

Notice that the denominator is $n-1$ to exclude the case where $i=j$ with related terms equal to zero.

It follows from Equations (2.2.7), (2.2.5), (2.2.8), and (2.2.9) that the total energy used in data collection is expressed as follows.

$$\mathcal{E}_h(k) = E_g + bkD + k(\bar{E} + \bar{e}). \quad (2.2.11)$$

The main issues involved in the optimal clustering model considered in this work are (i) finding the optimal number of clusters, (ii) selection of the optimal cluster heads/sinks, and (iii) associating the cluster members with the sinks. These issues can be solved by three algorithmic solutions: (a) a myopic k-means clustering algorithm where the optimal number of clusters $k = \mathcal{K}_{opt}$ is computed and the classical K-means algorithm is applied with $k = \mathcal{K}_{opt}$, (b) an optimized k-means clustering algorithm where the optimal number of clusters $k = \mathcal{K}_{opt}$ is computed and the \mathcal{K}_{opt} best cluster heads are selected and fed to the k-means algorithm to guide the clustering process, and (c) a multi-step clustering algorithm where a sequence of cluster head selection and cluster member association is performed on the network until all the nodes are assigned a cluster head or member status. Note that while the k-means algorithm can be applied to a dense and uniform network where each sensor node is able to communicate with its neighbors (see [1]), the multi-step algorithm would be more suitable for general networks where the connectivity property may not be met.

2.2.2 Problem Definition. The network considered in this chapter is denoted by $\mathcal{H}(\mathcal{N}, \mathcal{P}_g, \mathcal{P}_a, \mathcal{E}_g, \mathcal{E}_a)$, where \mathcal{N} is the set of sensor nodes and the UAVs' base stations' locations, \mathcal{P}_g is the set of paths expressing possible sensors communication pathways in the ground-based terrestrial network, \mathcal{P}_a the

set of paths in the airborne network consisting of possible routes followed by the UAVs to collect data delivered by the ground-based sensor network, and the energy consumed by the set of paths \mathcal{P}_g and \mathcal{P}_a is respectively represented by \mathcal{E}_g and \mathcal{E}_a . Given a hybrid network \mathcal{H} , the problem consists of finding the nodes' partition $\mathcal{P}(N)$ to minimize the total energy \mathcal{E}_h (see Equation (2.2.7)), such that each partition (cluster) is connected and its optimum head is known. The energy \mathcal{E}_h is referred to as the clustering cost. The network design consists of finding a network configuration that minimizes the clustering cost function subject to node selection and topology constraints with the objective of partitioning the network into two sets: a dominating¹ set of UAV collection points and a dominated set of cluster members forming the edge of the network. Mathematically formulated, the design process consists of finding a network partition \mathcal{C} derived from the graph of the type explained in Figure 2.2b, which leads to the optimal energy consumption \mathcal{E}_{opt} , such that \mathcal{N} is divided into disjoint clusters, where the cluster head is communicatively connected with all its cluster mates.

$$\mathcal{E}_{opt} = \min E_h = \min(\alpha E_g + \beta E_u + \gamma E_c) \quad (2.2.12)$$

Subject to,

$$\forall c \in \mathcal{C}, \exists x \in c, \forall y \in c, (x, y) \in P_g \quad (2.2.13a)$$

$$c_1, c_2 \in \mathcal{C}, c_1 \cap c_2 = \emptyset \quad (2.2.13b)$$

$$\bigcup_{c \in \mathcal{C}} c = N \quad (2.2.13c)$$

where, Constraints 2.2.13a shows the dominating set property of the set of cluster heads and 2.2.13b and 2.2.13c represent the network partitioning properties.

2.3 The Proposed Clustering Models

Two clustering algorithms were developed:

1. The UAV-Aware k-Means (UAKM) algorithm, which computes the number k of optimal clusters for hybrid dense networks to support/complement k-means clustering. Here, the number k is calculated using both the ground and the aerial networks, and hence, it considers the movement of the UAV.
2. The UAV-Aware DCC (UADC) algorithm, which adapts the DCC algorithm to include the UAVs data collection process.

2.3.1 The UAV-Aware K-Means Algorithm. In this subsection, we express the forms of energy in terms of the number of clusters k a hybrid network (see Section 5.2.2) needs to be partitioned into and use calculus to compute the value k that minimizes the total energy required for data collection. Energy Equation (2.2.11) can be expressed in terms of the number of clusters k , which in turn can be used to determine the optimal number of clusters, as shown in Equation (2.3.1).

¹In graph theory, a dominating set for a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one member of D . The domination number ($\gamma(G)$) is the number of vertices in a smallest dominating set for G [84].

$$\frac{\partial E}{\partial k} = E_e(a-2) + \frac{L^2 e_f(k-n)}{3k^2} - \frac{L^2 e_f}{3k} + bD + \bar{E} + \bar{e} \quad (2.3.1)$$

By solving the equation, $\frac{\partial \mathcal{E}_h}{\partial k} = 0$, where $k \geq 1$, we obtain the optimal value of \mathcal{E}_h for:

$$\mathcal{K}_{opt} = \sqrt{\frac{L^2 e_f n}{3(E_e(a-2) + bD + \bar{E} + \bar{e})}} \quad (2.3.2)$$

Notice that the second derivative is:

$$\frac{\partial^2 \mathcal{E}_h}{\partial k^2} = -\frac{2L^2 e_f(k-n)}{3k^3} + \frac{2L^2 e_f}{3k^2}. \quad (2.3.3)$$

We know that all observables in Equation (2.3.3) are positively valued. Furthermore, the difference $k-n$ is always negative (the number of cluster heads cannot exceed the number of all existing nodes). It follows that,

$$\frac{\partial^2 \mathcal{E}_h}{\partial k^2} \geq 0. \quad (2.3.4)$$

This confirms that the the total energy $\mathcal{E}_h(k)$ is a minimum at \mathcal{K}_{opt} , as shown in Equation (2.3.2).

Since the optimal number of clusters has to be a positive integer, the optimal number of clusters is denoted by \mathcal{K} , and it is calculated as follows.

$$\mathcal{K}_{opt} = \begin{cases} \lceil k \rceil & \text{if } E(\lceil k \rceil) \leq E(\lfloor k \rfloor) \\ \lfloor k \rfloor & \text{otherwise} \end{cases} \quad (2.3.5)$$

Consider three networks on which the following parameters (Referring to Equations 2.2.6 and 2.2.11) are defined in Table 2.1. The corresponding graphs of the energy are shown in Figure 2.3.

Parameter	Network 1	Network 2	Network 3	Units
n	100	100	200	
E_e	20	20	20	nJ/bit
E_p	21	21	21	nJ/bit/signal
e_f	1	1	1	pJ/bit/m ²
L	30	60	30	m
a	0.0008	0.0008	0.0008	
b	9	9	9	
D	6	6	6	
$\bar{E} + \bar{e}$	3	3	3	nJ/bit/signal

Table 2.1: Parameters and their corresponding values.

In Table 2.1, $\bar{E} + \bar{e}$ can be set to zero if we need to consider a case where some sensors are located together with refueling/repairing stations, which increase the energy of a UAV even if it were collecting data.

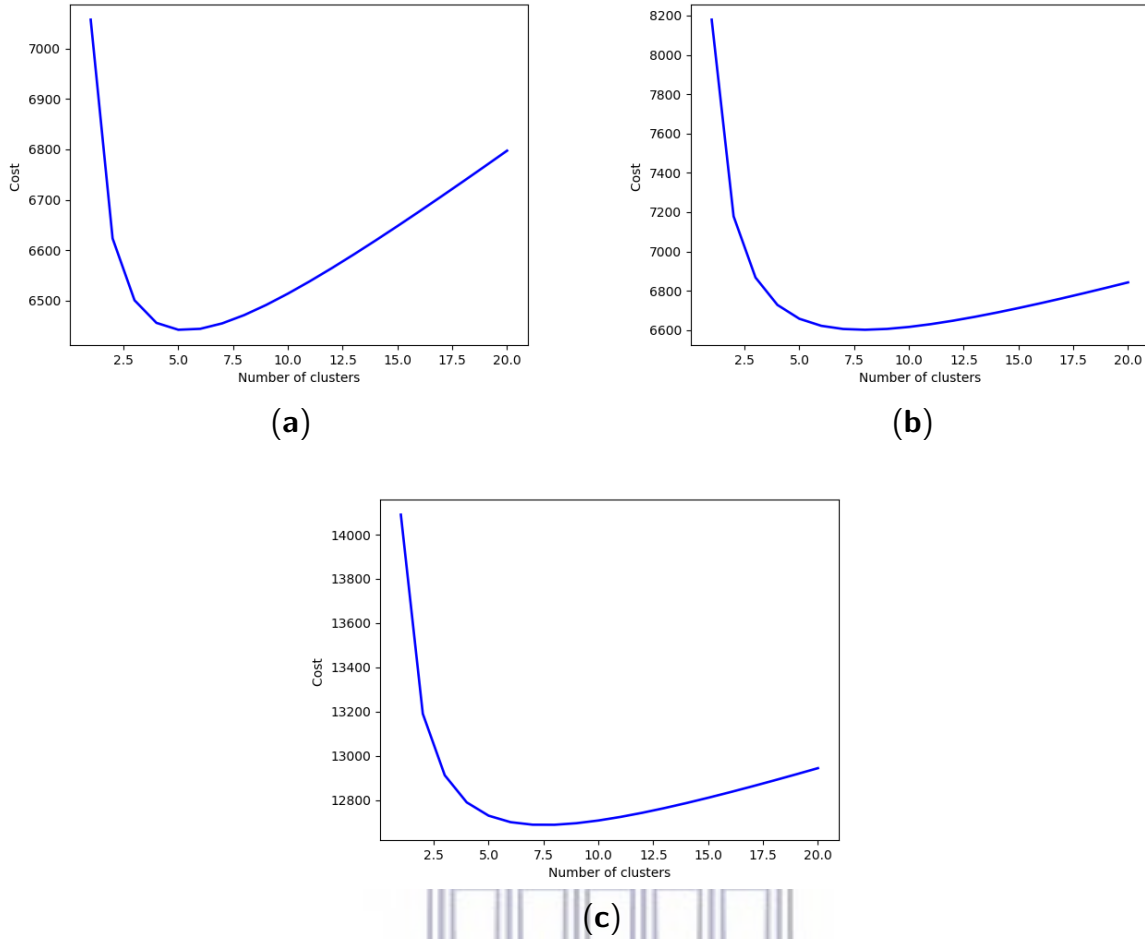


Figure 2.3: Energy required versus the number of clusters: (a) $30m \times 30m$ network with 100 nodes ; (b) $60m \times 60m$ network with 100 nodes; (c) $30m \times 30m$ network with 200 nodes.

Figure 2.3, showing the energy required compared to the number of clusters, reveals the optimal number of clusters for the three different networks. Figure 2.3a–c shows that the optimal number of clusters increases with the network size. Figure 2.3a shows that the number \mathcal{K}_{opt} for the first network (Network 1) lies in the interval $(4, 5)$. On the other hand, using Equation (2.3.2), $\mathcal{K}_{opt} = 4,84$. It follows from Equation (2.3.5) that,

$$\mathcal{K}_{opt} = \begin{cases} 5 & \text{if } E(5) \leq E(4) \\ 4 & \text{otherwise.} \end{cases} \quad (2.3.6)$$

Thus, the optimal number of clusters in this case is $\mathcal{K}_{opt} = 5$. Similarly, it can be shown that for the second network (Network 2),

$$\mathcal{K}_{opt} = \begin{cases} 6 & \text{if } E(6) \leq E(5) \\ 5 & \text{otherwise.} \end{cases}, \quad (2.3.7)$$

leading to $\mathcal{K}_{opt} = 6$. For the third network (Network 3),

$$\mathcal{K}_{opt} = \begin{cases} 7 & \text{if } E(7) \leq E(6) \\ 6 & \text{otherwise.} \end{cases}, \quad (2.3.8)$$

leading to a value of $\mathcal{K}_{opt} = 7$.

2.3.2 The UAV-Aware DCC Algorithm. The UADC algorithm has been designed based on a multi-step process using the following cluster head selection assumptions:

- Density-aware selection policy, where nodes are assigned the cluster head identity based on their node degree $deg(i)$. While leading to the UAV choosing data collection points with a high volume of data, this policy might lead to the UAV flying longer distances to collect these data, and hence, depleting its energy during its inbound journey.
- Distance-aware selection policy, where nodes are elected cluster heads based on the expected Dijkstra's shortest path from the nodes to all other nodes, following the links in the airborne network (links of the network \mathcal{H}_a). This policy aims at minimizing the energy usage of the airborne sensor network, but might lead to the UAV being tasked with collecting data at collection points with very few data.
- A hybrid policy that combines features from dense and distance-aware cluster head selection by combining both parameters into a weighted sum metric expressed by:

$$P(i) = \lambda deg(i) + \psi \frac{1}{D_i}. \quad (2.3.9)$$

Here, $deg(i)$ represents the number of available neighbors (of node i) in the network of type \mathcal{H}_a , whereas D_i is the average distance from node i to all nodes in the network of type \mathcal{H}_g . λ and ψ are coefficients corresponding to the node degree in \mathcal{H}_g and average distance in \mathcal{H}_a , respectively.

This policy is used in clustering as shown by the proposed algorithm described as follows.

Input: The graph of type $H(\mathcal{H}_a, \mathcal{H}_g)$

Output: A dictionary of cluster heads and their cluster mates

In Algorithm 1, the first steps consist of computing a list L of all link lengths in the SN (network of type \mathcal{H}_g) and the dictionary D_P , whose keys are the sensor labels, and the corresponding values consist of the average distance to each node in the restricted network (network of type \mathcal{H}_a). The minimum coverage energy E_{min} is initialized to infinity. The network clustering is expressed in the form of a dictionary whose keys are the cluster heads, and the values correspond to the clusters' members. The clusters' dictionary C is initially set to empty (Line 4). The cluster dictionary is assumed to have the cluster heads as keys, and their corresponding values are the list of nodes each cluster head is to support.

Algorithm 1: Optimal clustering.

```

1  $L \leftarrow$  set of the lengths of links in  $L_g$ 
2  $D_p \leftarrow$  dictionary of nodes (keys) and their expected length of the shortest path to each sensor node in  $H_a$  (values).
3  $E_{min} \leftarrow \infty$ 
4  $C_{min} = \{\}$ 
5 for  $Radius \in L$  do
6    $N_{rad} \leftarrow$  dictionary of nodes (keys) and a list of their available neighbors at distance  $dist \leq Radius$ 
7   Order  $N_{rad}$  in terms of the decreasing order of the value of the price  $P$  of the keys (cluster heads)
8    $Ch \leftarrow$  List of  $N_{rad}$  ordered keys (cluster heads)
9    $Cv \leftarrow$  List of  $N_{rad}$  ordered values (cluster mates)
10   $C \leftarrow$  empty dictionary, which will contain clusters
11  while  $Ch \neq \emptyset$  do
12     $C_{Ch_0} \leftarrow Cv_0$ 
13    Remove  $Ch_0$  and all nodes in  $Cv_0$  from  $Ch$ .
14     $N_{rad} \leftarrow$  dictionary of nodes in  $Ch$  (keys) and a list of their neighbors not in any formed clusters
15    Order  $N_{rad}$  in terms of the value of the price  $P$  of the keys (cluster heads)
16     $Ch \leftarrow$  List of  $N_{rad}$  ordered keys (cluster heads)
17     $Cv \leftarrow$  List of  $N_{rad}$  ordered values (cluster mates)
18  end
19  Calculate the price  $P(C)$  using Equation (3.3)
20  if  $P(C) < E_{min}$  then
21     $C_{min} \leftarrow C$ 
22     $E_{min} \leftarrow P(C)$ 
23  end
24 end
25 Return  $C_{min}$ 

```

- **Complexity:** Line 5 shows that the time complexity depends on the total number \mathcal{L} of links and both Lines 6 and 11 show that the time complexity depends also on the total number n of the nodes in the network. Since Line 11 shows that there is an inner loop, it is clear that the worse case scenario would reveal that the time complexity for the algorithm is $\mathcal{O}(n \cdot \mathcal{L})$.

From Line 5 on, each link length (Euclidean distance between two connected nodes) is used as the clustering radius (maximum distance of nodes and cluster heads), to form a corresponding clustering C .

Clustering is done using a dictionary N_{rad} , consisting of nodes and their \mathcal{H}_g neighbors at a distance less than or equal to the chosen radius. Note that the radius is only chosen from a list of lengths of the \mathcal{H}_g links, and it is assumed to be the same for all clusters to be formed. This dictionary gets formed (Line 6), and using the pricing shown by Equation (3.3), it is decreasingly ordered (Line 7).

Let Ch be a list of ordered N_{rad} keys (list of potential cluster heads) and Cv be the list of the corresponding keys (possible cluster members). To form the first cluster, we take the first element of the list Ch to be the cluster head, and the first list in Cv constitutes the corresponding cluster mates.

N_{rad} is then updated to contain the remaining possible cluster heads and their neighbors, which are the nodes not in any of the formed clusters.

The process of using the available nodes in the list N_{rad} to make one cluster is repeated (Line 11–16) until no more cluster heads are available. In this case, one clustering configuration is done, and its cost is computed using Equation (5.2.3) (Line 17).

Each new clustering-related cost is compared to the existing minimum cost to check the possibility of updating the best cluster C_{min} and the corresponding cost E_{min} .

An example that shows graphically how Algorithm 1 works is presented below. For simplicity, only the network of type \mathcal{H}_g is shown. The considered cluster radius is assumed to be the maximum link length, and hence, cluster heads will be associated with all their neighbors in \mathcal{H}_g .

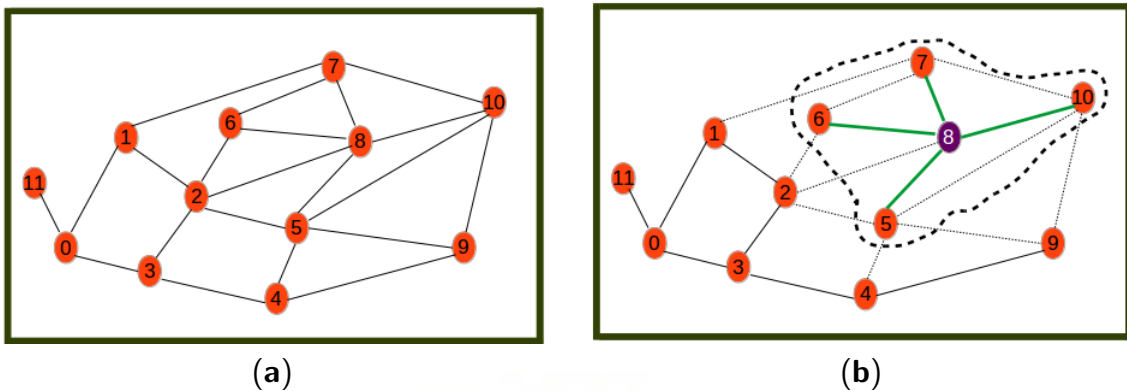


Figure 2.4: Beginning steps. (a) **Step 0**: initial network; (b) **Step 1**.

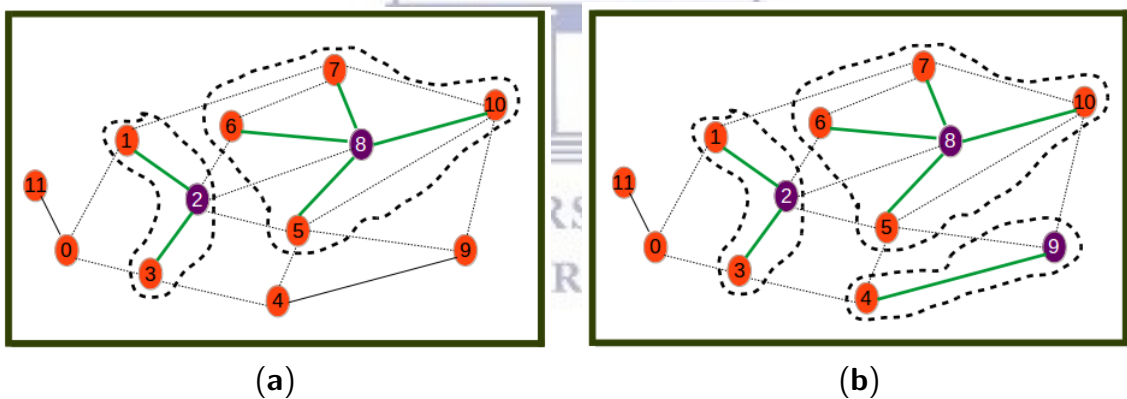


Figure 2.5: Processing steps. (a) **Step 2**; (b) **Step 3**.

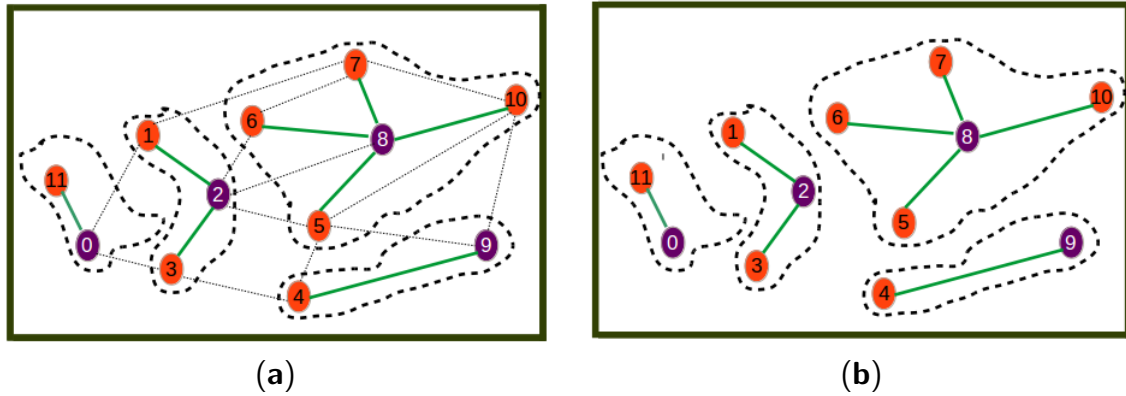


Figure 2.6: Last steps. (a) **Step 4**; (b) **Step 5**.

Figures 2.4, 2.5, 2.6 show the different steps involved in the clustering algorithm example and are explained below:

Step 0 is the initial step revealing the initial network.

Step 1 selects Node 8 as the one with highest utility (computed by Equation (3.3)) to become the cluster head. The first cluster is formed by assigning all its neighbors as its cluster members.

Step 2 selects Node 2 as the next best cluster node. Here, to calculate the utility, the nodes or links involved in the formed cluster are not considered. This is why for example Node 2 has a new degree of two. The new degree of Node 0 is greater than that of Node 2 even though the utility of Node 2 is highest since it is the closest node to the remaining nodes.

Step 3 is a step where Node 9 is selected as the next best cluster head, which is joined by only Node 4 as its cluster member.

Step 4 is a step where Node 0 is selected as the last best cluster head, which is joined by Node 11 as its neighbor to form the last cluster.

Step 5 is the last phase where the resulting cluster and the corresponding communication links through which nodes have to send sensor readings to cluster heads are shown.

Proposition.

Algorithm (1) satisfies the following properties.

- P1. The produced cluster heads constitute a dominating set of the network of type \mathcal{H}_g (see Constraint 2.2.13a).
- P2. The set C of the produced clusters is a partition of the set of all nodes (see 2.2.13b and 2.2.13c).

proof.

- P1. **Dominating set property:** Lines 6, 8, and 11 show that at the end, each node becomes either a cluster head or a cluster member. On the other hand, Lines 1, 2, 9, and 6 show that only neighbors

of the cluster head are added in the same cluster to be cluster members. It then follows that when the algorithm halts, if a node is not a cluster head, then it is connected to the cluster head in the same cluster.

- P2. **Partition property:** Line 13 shows that no node (cluster member or cluster head) belongs to more than one cluster. Hence, the formed clusters are mutually exclusive. On the other hand, Lines 6, 8, and 11 show that the algorithm halts when each node has either been a cluster member or (exclusively) a cluster head. This shows that in the end, each node belongs in a unique cluster. Hence, cluster nodes constitute a partition of the ground network nodes.

2.4 Issues and Relaxation

In this section, we discuss two main issues and address them to improve the performance of the algorithm.

2.4.1 Energy Inefficiency. The proposed algorithm is greedy in terms of the way cluster members are assigned to cluster heads. The assignment of all neighbor nodes (at a distance less than or equal to a threshold) to cluster heads does not necessarily lead to the best association between cluster members and cluster heads. This may lead to the case where nodes are assigned to cluster heads that are not closest to them. This would result in higher energy/cost for data aggregation on cluster heads. This issue is depicted in Figure 2.7a.

Figure 2.7a shows an inefficient clustering where Node 3 has been allocated as the cluster member of Node 0 instead of Node 4, which is the closest cluster head. Figure 2.7b reveals that through relaxation, nodes re-choose their corresponding clusters depending on their closest elected cluster heads, thus leading Node 3 to become a cluster member of Node 4. The solution to the energy inefficiency issue above consists of applying the relaxation algorithm below to improve the UAKM and UADC algorithms.

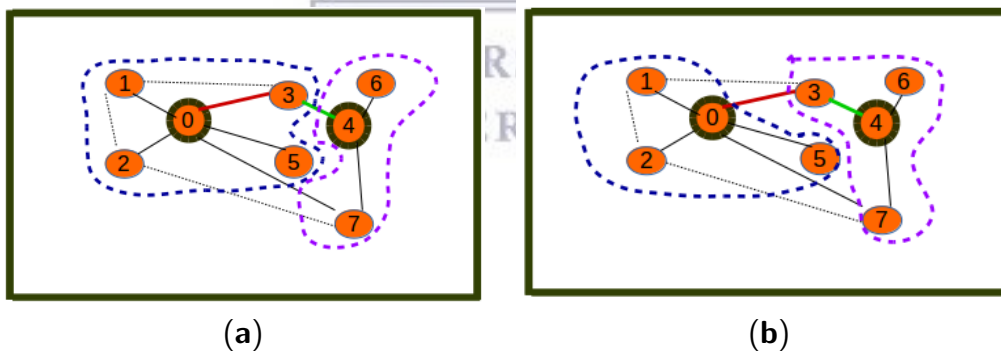


Figure 2.7: Relaxation: energy inefficiency. (a) Non-relaxed clustering; (b) Distance-aware clustering.

Input:

- The graph of type $H(\mathcal{H}_a, \mathcal{H}_g)$.
- the initial clustering (using Algorithm 1): each node and its initial cluster head denoted by n and n_{ch0} , respectively.

Output:

- ← A more efficient clustering.

Denote n_{ch} the new cluster head of node n . Assume C is the set of all cluster heads and N is the set of all cluster members (all the \mathcal{H} nodes excluding cluster heads).

Algorithm 2: Distance-aware node redistribution.

```

1                                     ▷ Loop to allocate each node a cluster head
2 for  $n \in N$  do
3    $n_{ch} \leftarrow n_{ch0}$ 
4                                     ▷ Loop to compute the closest cluster head to node  $n$ 
5   for  $c \in C$  do
6     if  $d(n, c) < d(n, n_{ch})$  and  $(c, n) \in \mathcal{P}_g$  then
7        $n_{ch} \leftarrow c$ 
8     end
9   end
10 end

```

As calculated the time complexity for Algorithm 1, It is clear that the time complexity of Algorithm 2 is $\mathcal{O}(n \cdot \mathcal{C})$, where \mathcal{C} is the size of the set C .

2.4.2 Orphan Nodes. The presence of orphan nodes leading to isolated cluster heads is another issue of the proposed greedy algorithm that can reduce the utility of the hybrid network as it can lead to the UAV being tasked to collect data on a cluster-head with very reduced data. This is illustrated by Figure 2.8a, which reveals a sensor network with three clusters: the first cluster with Node 0 as the cluster head and Nodes 1, 2, and 5 as cluster members, the second with Node 4 as the cluster head and Nodes 6 and 7 as cluster members, and the last cluster, which has the orphan Node 3 as the isolated cluster head. By applying a distance-aware node redistribution process, the sensor network will be restructured into a two-cluster network similar to the one depicted by Figure 2.8b with two clusters: the first cluster with Node 0 as the cluster head and Nodes 1, 2, and 5 as cluster members and the second cluster with Node 4 as cluster head and Nodes 3, 7, and 6 as cluster members.

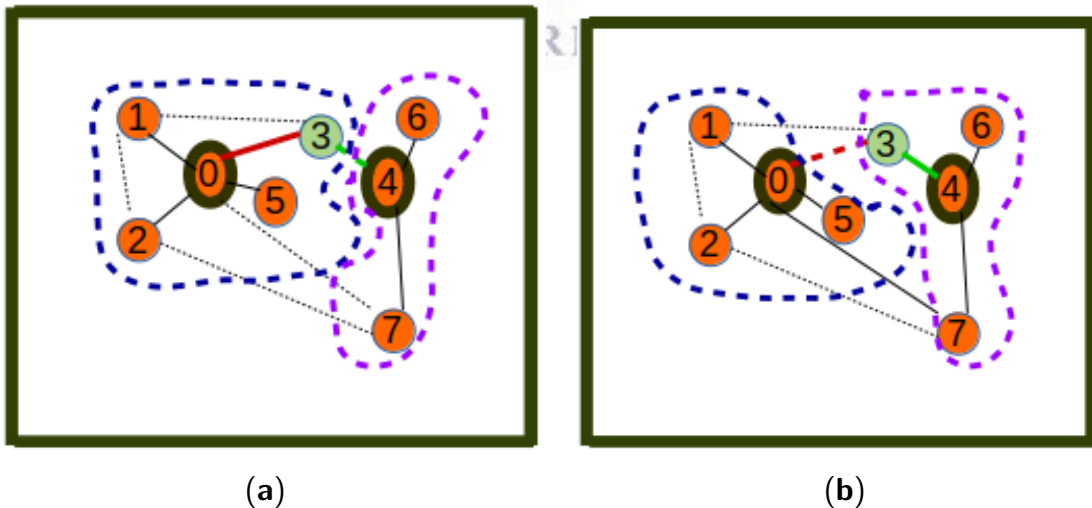


Figure 2.8: Relaxation: orphan nodes. (a) Non-relaxed clustering; (b) relaxed clustering.

Figure 2.8a reveals a clustering where Node 3 is an orphan node in a cluster consisting of only one cluster head with no cluster member, while Figure 2.8b reveals a situation where the orphan cluster head is assigned to the optimal cluster whose cluster head is nearest to the orphan node, thus becoming one of its cluster members. The solution to the orphan node inefficiency issue related to the above consists

of applying the cluster restructuring algorithm below to improve the UAKM and UADC algorithms. Note that while this algorithm is based on the same principles as the distance-aware node redistribution, cluster restructuring may require balancing the benefits due to energy efficiency and the data muling utility in order to decide on whether to move an orphan node into another cluster to become a cluster member or leave the orphan node in its current cluster.

Input:

- The graph of type $H(\mathcal{H}_a, \mathcal{H}_g)$.
- the initial clustering (using Algorithm 1): each node and its initial cluster head denoted by n and n_{ch0} , respectively.

Output:

- ← Distance-aware clustering.

Denote n_{ch} as the new cluster head of node n and $ut(n)$ a Boolean value indicating if it is more beneficial to restructure the network. The re-clustering may be required when the cost of sending data to n_{ch} is smaller than the cost of visiting the node with a UAV.

Assume C is the set of all cluster heads and N is the set of all cluster members (all the \mathcal{H} nodes excluding cluster heads).

Algorithm 3: Distance-aware cluster restructuring.

```

1                                     ▷ Loop to allocate each node a cluster head
2 for  $n \in N$  do
3    $n_{ch} \leftarrow n_{ch0}$ 
4                                     ▷ Loop to compute the closest cluster head to node  $n$ 
5   for  $c \in C$  do
6     if  $d(n, c) < d(n, n_{ch})$  and  $ut(n) = 1$  and  $(c, n) \in \mathcal{P}_g$  then
7        $n_{ch} \leftarrow c$ 
8     end
9   end
10 end
```

Notice that the time complexity of Algorithm 3 is $\mathcal{O}(n \cdot \mathcal{C})$, where \mathcal{C} is the size of the set C (the same as Algorithm 2).

2.4.3 The Update Step. As suggested above, both the UAKM and UADC algorithms can be updated into a two-step algorithm that applies the basic algorithm first (UAKM or UADC) and thereafter balances the network using the distance-aware relaxation algorithm above. We adapt the k-means update step to achieve energy efficiency by using the fact that the knowledge of the cluster heads can help redistribute cluster members according the closeness to cluster heads, as shown in Figure 2.7b. The same applies to the UADC algorithm, which is complemented by a relaxation step to balance the energy consumption as suggested above.

2.4.4 Remark.

- The distance-aware relaxation algorithm proposed above may lead to energy consumption improvement.

- The restructuring of the terrestrial sensor network is another relaxation technique that follows the same distance-aware strategy for a different purpose, but it can also lead to energy consumption improvement.

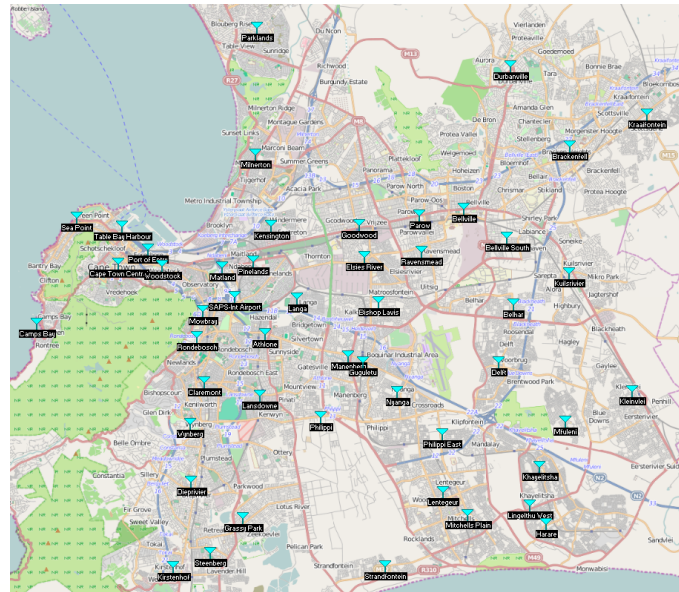
2.5 Results and Discussion

In this section, we report on the experimental results obtained from running the proposed algorithms in different settings. The algorithms (both UADC and relaxed UADC) are analyzed and compared to the DCC and UAKM for benchmarking purposes. We considered two network topologies: (i) a random network and (ii) the city of Cape Town network used as a smart city use-case.

2.5.1 Smart City Use-Case. We considered the public safety network topology consisting of Cape Town (South Africa) police stations as collection points of the terrestrial/ground sensor network. This network was used as a smart city use-case aiming to provide citizen safety and city surveillance through a combination of aerial and terrestrial traffic control. The Cape Town police stations are labeled in terms of integers in the interval $[1, 49]$, and their GPS coordinates were used as their positions (see Figure 2.9a). The corresponding positions on a map are shown in Figure 2.9b. The Radio Mobile software [85] was used to create the hybrid network by having the terrestrial/ground communication network (see Figure 2.10a) generated using a two-step process consisting of (i) generation by the mobile radio of a terrestrial network that considers only connections whose link margin is greater than 50 dB in the white space spectrum frequency and (ii) generation by the mobile radio of an aerial network consisting of UAV paths (see Figure 2.10b) that considers only connections/links with a link margin between 30 and 50 dB in the same white space band. For the sake of clarity of links, the topology on the map does not necessarily conform with the processed network.

2.5.2 Hybrid Clustering: $\alpha = \beta = \gamma = 10$. We conducted the first experiment to study the impact of the radius and number of clusters on the performance of a hybrid clustering model where both layers were considered: airborne and terrestrial network by setting the clustering parameters to $\alpha = 10$ and $\beta \neq 0$ and $\gamma \neq 0$. This parameter setting corresponds to the case where the energy spent by the UAVs is considered and relate energy consumption has been taken to be of the same importance. The results presented in Figure 2.11a show that the coverage cost (total coverage energy) reduced with the increase of the radius following a logarithmic function that led to a convergence value that did not necessarily correspond to the optimal point.

Label	Station	Longitude	Latitude
1.	Bellville	33°54'26"S	018°37'12"E
2.	Nyanga	33°59'29"S	018°34'54"E
3.	SAPS-Int Airport	33°56'37"S	018°29'18"E
4.	Bellville South	33°54'55"S	018°38'40"E
5.	Philippi	34°00'02"S	018°32'16"E
6.	Matland	33°55'46"S	018°28'52"E
7.	Parow	33°54'16"S	018°35'39"E
8.	Lansdowne	33°59'26"S	018°39'10"E
9.	Rondebosch	33°57'46"S	018°27'59"E
10.	Kuilsrivier	33°55'56"S	018°40'50"E
11.	Wynberg	34°00'15"S	018°27'46"E
12.	Sea Point	33°54'20"S	018°33'51"E
13.	Bishop Lavis	33°56'46"S	018°34'15"E
14.	Cape Town Central	33°55'40"S	018°25'16"E
15.	Table Bay Harbour	33°54'36"S	018°25'24"E
16.	Delft	33°58'29"S	018°38'24"E
17.	Mowbray	33°57'00"S	018°28'12"E
18.	Ravensmead	33°55'19"S	018°35'45"E
19.	Goodwood	33°54'33"S	018°33'35"E
20.	Elsies River	33°55'28"S	018°33'46"E
21.	Port of Entry	33°55'15"S	018°26'18"E
22.	Kensington	33°54'34"S	018°30'33"E
23.	Athlone	33°57'41"S	018°30'20"E
24.	Woodstock	33°55'43"S	018°28'48"E
25.	Pineclands	33°55'36"S	018°29'66"E
26.	Belhar	33°56'49"S	018°38'55"E
27.	Manenberg	33°58'18"S	018°33'13"E
28.	Claremont	33°59'04"S	018°28'15"E
29.	Guguletu	33°58'29"S	018°33'43"E
30.	Durbanville	33°59'01"S	018°38'49"E
31.	Brackenfell	33°52'18"S	018°40'51"E
32.	Langa	33°58'39"S	018°31'27"E
33.	Mitchells Plain	34°02'51"S	018°37'49"E
34.	Khayelitsha	34°01'29"S	018°39'48"E
35.	Huleni	34°00'11"S	018°40'43"E
36.	Lingelitsh West	34°02'35"S	018°39'28"E
37.	Diepsrivier	34°01'54"S	018°27'47"E
38.	Kleinrivier	33°59'18"S	018°43'00"E
39.	Harare	34°03'06"S	018°40'01"E
40.	Grassy Park	34°02'55"S	018°29'35"E
41.	Steenberg	34°03'50"S	018°28'20"E
42.	Kirstenhof	34°04'19"S	018°27'11"E
43.	Camps Bay	33°57'22"S	018°22'29"E
44.	Wiltonton	33°52'32"S	018°30'01"E
45.	Parklands	33°48'55"S	018°30'04"E
46.	Kraaifontein	33°51'24"S	018°43'30"E
47.	Philippi East	34°00'32"S	018°36'27"E
48.	Strandfontein	34°04'19"S	018°34'29"E
49.	Lenteguur	34°02'11"S	018°36'29"E



(a) GPS positions; (b) positions on the map.

Figure 2.9: Case study. (a) GPS positions; (b) positions on the map.

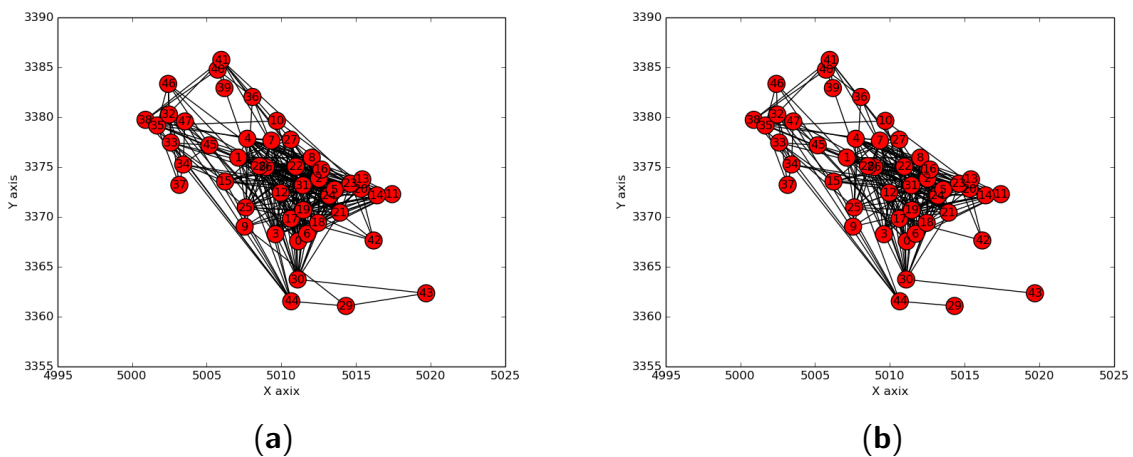
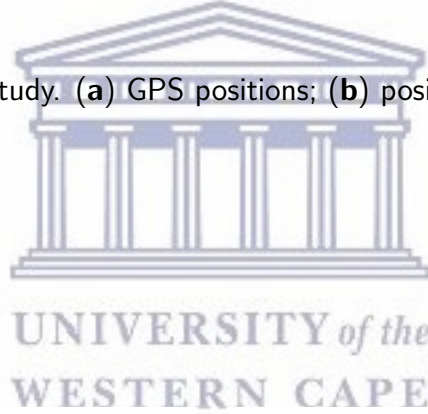


Figure 2.10: Processed networks . (a) Communication network considered; (b) UAV path network considered.

On the other hand, the results presented in Figure 2.11b reveal a different trend where the coverage cost (total coverage energy) increases linearly with the increase of the number of clusters. These results are in line with those presented in Figure 2.11a, since a lower radius will logically lead to a higher number of clusters and subsequent higher cost, while a higher radius will logically lead to the algorithm finding a lower number of clusters and subsequent lower coverage cost resulting from a high transportation cost. The best clustering would then be the one related to the radius, which minimizes the number of clusters and hence leads to the minimum transportation cost. In Figure 2.11a, such an optimal radius is 13, and it corresponds to four clusters and a transportation cost of close to 25 joules.

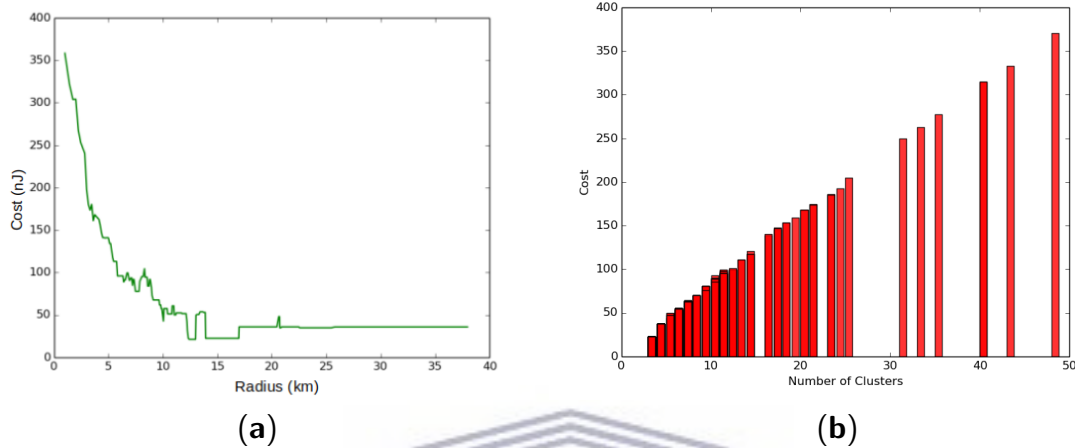


Figure 2.11: Impact of parameters on performance. (a) Cost versus radius; (b) coverage cost versus the number of clusters.

2.5.3 Terrestrial Clustering: $\alpha = 10$ and $\beta = 0$ and $\gamma = 0$. We conducted a second experiment to evaluate the impact of the UAV presence on the hybrid clustering process by setting the parameters $\alpha = 10$ and $\beta = 0$ and $\gamma = 0$, which represent a setting where only the energy consumed for transmission and reception in the ground/terrestrial network is considered, discounting the data muling energy consumed by the UAV.

The results presented in Figure 2.12a reveal a different trend compared to the hybrid network setting in Figure 2.11a where:

1. The coverage cost function increases with the increase in the radius size following an exponential function leading to a convergence value where the cost becomes constant.
2. The clustering process leads to much smaller coverage cost values (less than 1.0 joule) as compared to the general case where the coverage cost values ranged between 20 and 370 joules.

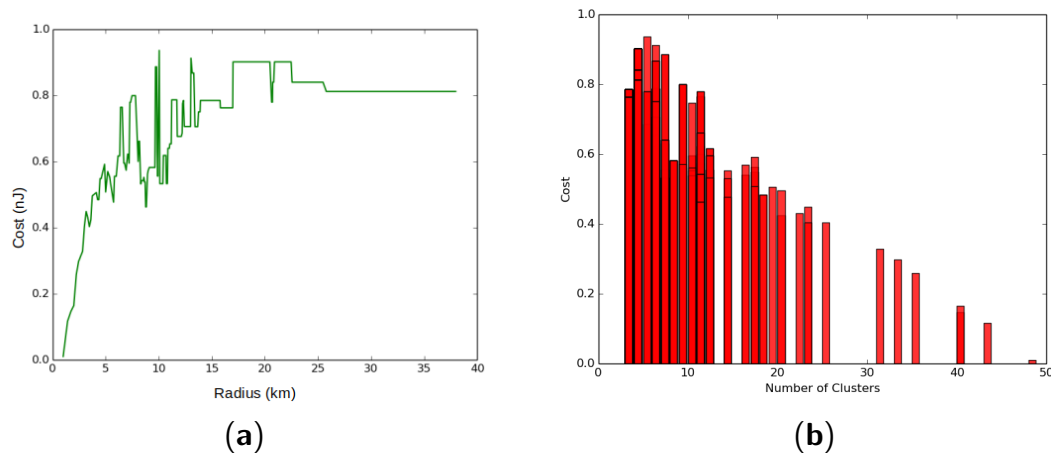


Figure 2.12: Impact of UAV on clustering. (a) Radius-cost; (b) communication network.

Similarly, Figure 2.12b shows a different trend compared to the hybrid clustering in Figure 2.11b, where the coverage cost decreases with the increase in the number of clusters, but not necessarily following a strict linear trend. The correlation between the values in Figures 2.11b and 2.12b is negative and smaller.

2.5.4 The Impact of the Cluster Head Selection Parameter on Performance. We conducted a set of experiments to evaluate the impact of the cluster head selection policy on performance by setting the parameters $\psi = 100$ and varying λ from 0 to 1 as follows (referring to Equation 3.3). We consider the following parameter setting (referring to policies in Section 2.3.2).

- $\lambda = 0$ expressing a distance awareness policy.
- $\lambda = 0.25$ expressing a balanced policy with a more focused distance awareness trend.
- $\lambda = 0.5$ expressing a fair, balanced policy between density and distance awareness.
- $\lambda = 0.75$ expressing a balanced policy with a more focused density awareness trend.
- $\lambda = 1$ expressing the density awareness policy.

The goal was to assess how the three different policies would impact the overall coverage cost.

The results presented in Figure 2.13 revealed that:

- Distance awareness decreases the total coverage cost more slowly than density awareness: at any given radius, the distance awareness policy cost is higher than the density awareness policy cost, as revealed by the red curve corresponding to $\lambda = 0$.
- Any balanced policy $0 < \lambda < 1$ leads to the same and lower energy cost as the density awareness policy $\lambda = 1$.

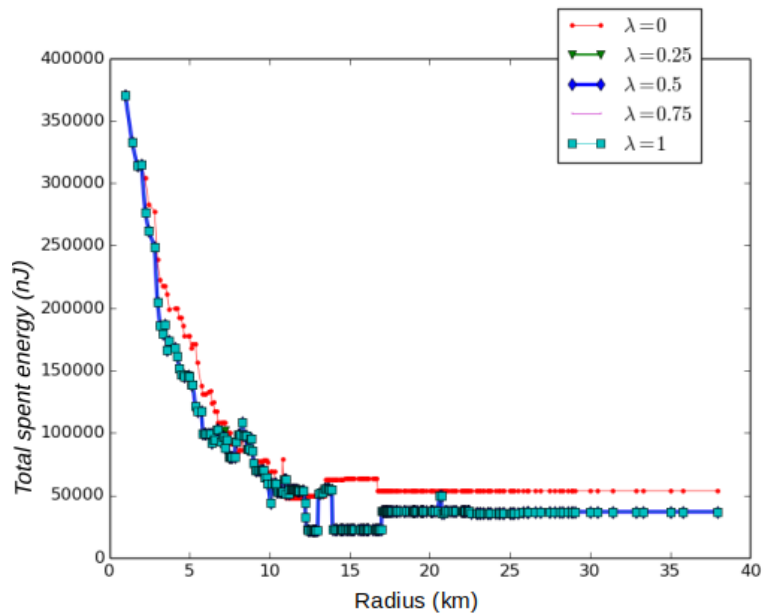


Figure 2.13: Impact of the cluster-head selection policy.

2.5.5 UADC versus DCC Performance Comparison. We conducted another experiment to compare the performance of the DCC algorithm (using only density awareness: $\lambda \neq 0$ and $\psi = 0$) to the UADC algorithm (using both density and distance awareness: $\lambda \neq 0$ and $\psi \neq 0$) using a variety of network topologies. The following five settings/cases (see Figure 2.14) were considered:

- Case 1: The UAV's paths constitute a proper sub-network of the terrestrial communication network.
- Case 2: The terrestrial communication network is a proper sub-network of the UAVs' network. This has been achieved by interchanging the networks chosen for the experiment in Figure 2.14a.
- Case 3: The two networks (terrestrial and aerial) are the same. Here, the assumed network is shown by Figure 2.10a.
- Case 4: In this experiment, positions were kept the same, and for both types of networks, the connections were generated randomly.
- Case 5: In this experiment, both the positions and links of both networks were generated randomly. The total number of considered nodes was still 48, but their positions were generated by randomly selecting the coordinates from a normal distribution with mean = 500 and a standard deviation of 300 ($\mathcal{N}(500, 300)$).

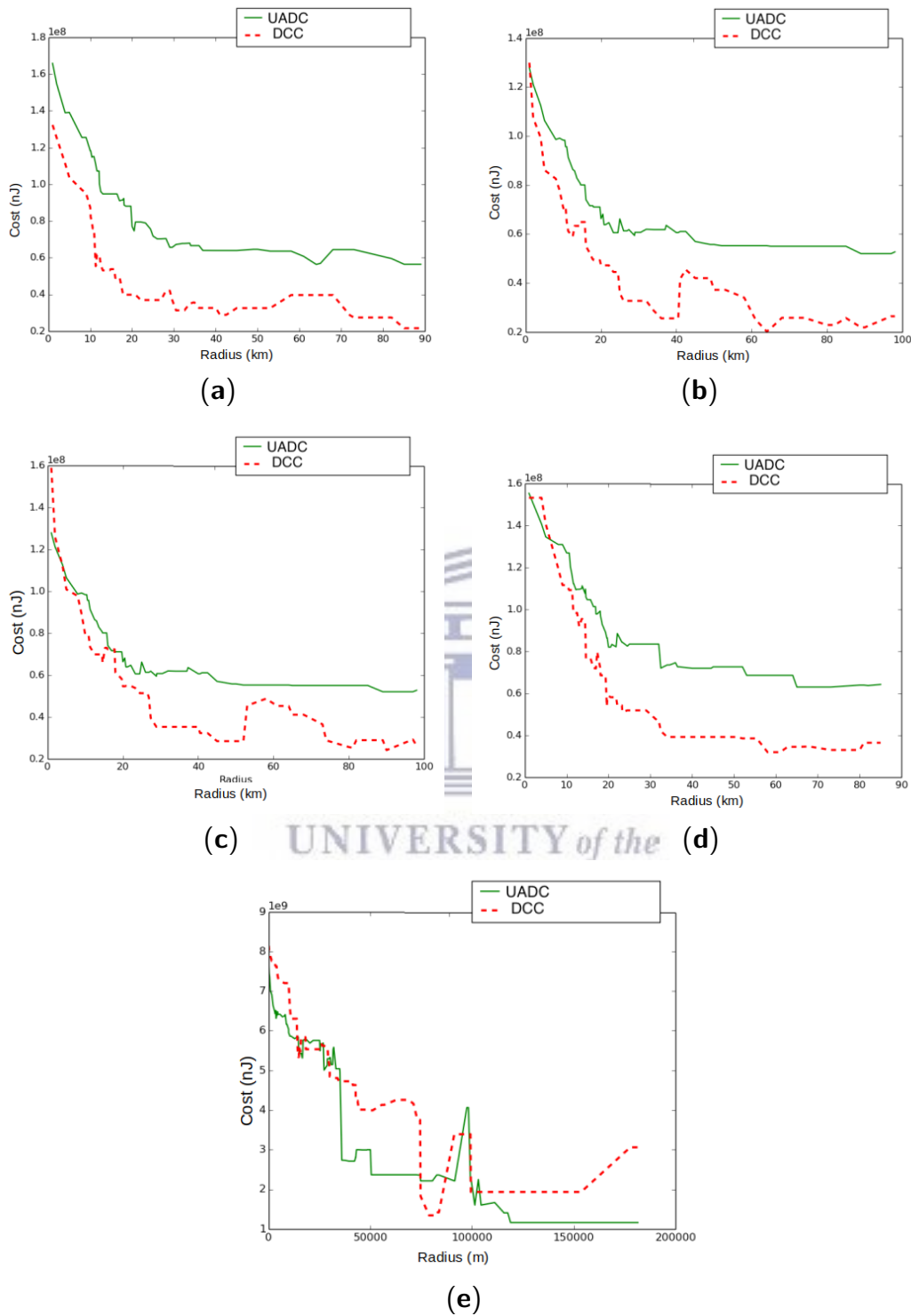


Figure 2.14: Algorithms comparison on different topologies. (a) Case 1; (b) Case 2; (c) Case 3; (d) Case 4; (e) Case 5. UADC, UAV-Aware DCC.

Figure 2.14 shows the UADC is outperformed by DCC when the radius is small ($< 100km$). However in case the radius is high ($> 100km$) Figure 2.14e shows that UADC outperforms the DCC. This highlights that there is no need to use UAVs in case clusters are not large enough.

Figure 2.15 reveals the total difference in coverage cost between the UADC and DCC algorithms as a function of the radius. These results reveal that:

- With the exception of Case 5, UADC leads to higher coverage cost compared to DCC as a result of the data muling cost due to the energy consumption of the UAV.
- The case where the terrestrial communication network is a proper sub-network of the aerial network (Case 2) leads to lower coverage cost compared to the reverse case (Case 1) where the aerial network is a sub-network of the terrestrial network.
- The lowest UADC cost is achieved when the aerial network and the terrestrial networks are the same (Case 3).
- The case where both networks have the same positions, but randomly-generated connections (Case 4) leads to higher coverage cost compared to the case where both networks are the same (Case 3) for both the UADC and DCC algorithms.
- The case where positions and links are randomly generated for both networks (Case 5) is the only case where the UADC algorithm outperforms the DCC algorithm for some of the higher radius sizes.

The proposed algorithm evolves. It shows that when the UAVs' paths constitute a sub-network of the communication network, the adoption of the DCC's policy was best for all the algorithm's steps (see Figure 2.15a). However, considering the converse case (Figure 2.15b), the UAV-aware policy outperformed the adopted DCC in only two cases, but the lowest energy corresponded to the adoption of the DCC. For the case in Figure 2.15d, we observe more cases where the UAV-aware policy was better than adopting the DCC, but still, the minimum energy corresponds to the DCC adoption. Randomly generating the nodes positions, Figure 2.15e shows that the UAV-aware policy was the one corresponding to the lowest energy and hence outperformed the adoption of the DCC.

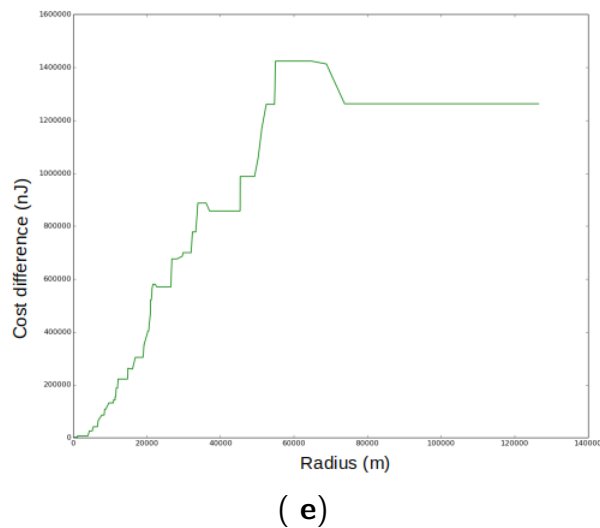
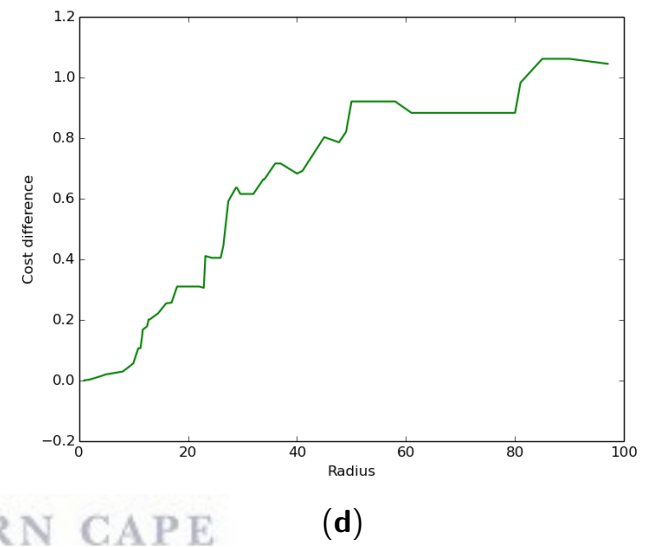
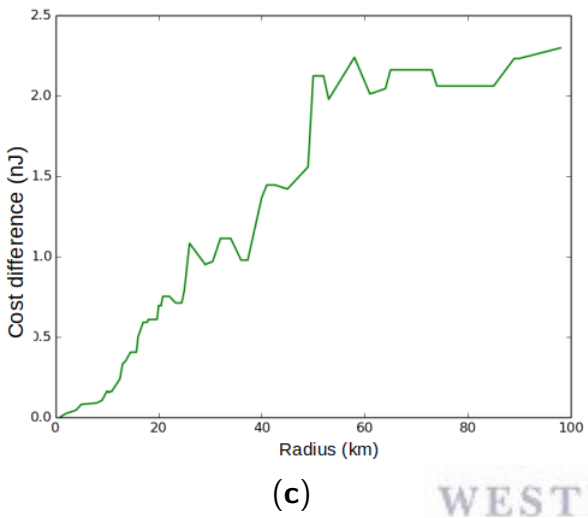
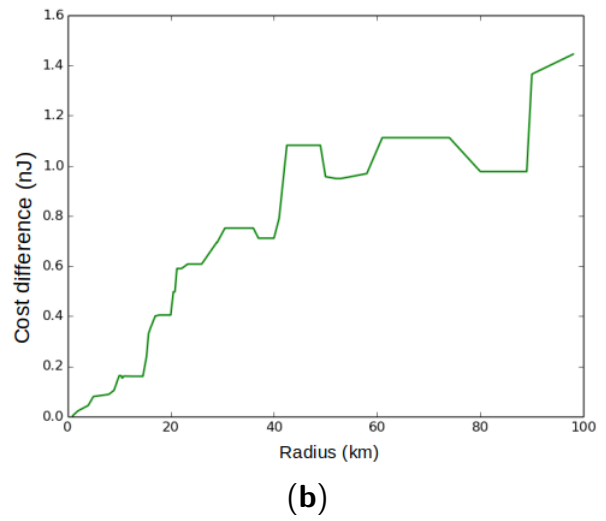
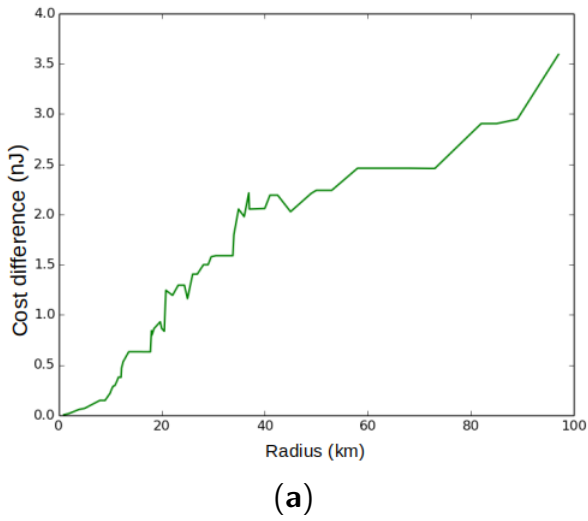


Figure 2.15: Algorithms difference based on different topologies. (a) Case 1; (b) Case 2; (c) Case 3; (d) Case 4; (e) Case 5.

Note fluctuation in both Figures 2.14 and 2.15. This is caused by the size of clusters. It is important to mention that UAVs usage in a small cluster (cluster of small radius) consumes more energy. However, if the radius is significantly large, the UADC is more efficient than DCC.

2.5.6 The Impact of Relaxation on Performance. In this subsection, we show the impact of the relaxation algorithm (see Algorithm 2) on performance by revealing the difference of the total coverage cost between the UADC algorithm without and with relaxation for the same five different cases described above.

The results presented in Figure 2.15 for all five cases reveal positive values for all radii. This reveals that the proposed distance-aware relaxation (and similarly, the distance-aware restructuring) had a positive impact on the performance achieved by the UADC algorithm. Furthermore, the figures reveal an increase of the coverage cost difference with the radius. The results also reveal a variation of such an increase with the cases where it is more pronounced for some cases compared to others, as shown by the slope and values of the different cost difference functions.

2.5.7 Reliability of the Family of k-Means Algorithms. In this subsection, we evaluate the connectedness of the network configuration, which expresses the reliability of the k-means and UAKM algorithms in terms of intra-cluster connectivity. The connectedness is a key property that determines the efficiency of the data muling process handled by the UAV in the hybrid network scenario since the sensor readings are collected by the moving UAV only when visiting cluster heads. Therefore, a highly-disconnected network will lead to high missing data. Note that a poorly-connected and less reliable network configuration will reveal lower intra-cluster connectivity, while a more reliable and highly-connected network configuration will result in higher intra-cluster connectivity. The results are shown in Figures 2.16a,b and in Table 2.2 in terms of average disconnectedness. Figure 2.16a and Table 2.2 reveal the results for the city of Cape Town network depicted by Figure 2.10a, while Figure 2.16b shows the results of a random network. The average disconnectedness was computed as the percentage of (orphan) nodes that have been assigned to clusters by the k-means algorithm, but that were not connected to related cluster heads. A hundred runs were performed for every run and every value k of the cluster with the number k of clusters ranging from one to the total number of nodes of the network. Figure 2.16b shows the average disconnectedness for a random 100-node network where the coordinates of the 100 nodes' positions were randomly chosen from a standard normal distribution of size 1000, and the links were also randomly generated to get a connected graph.

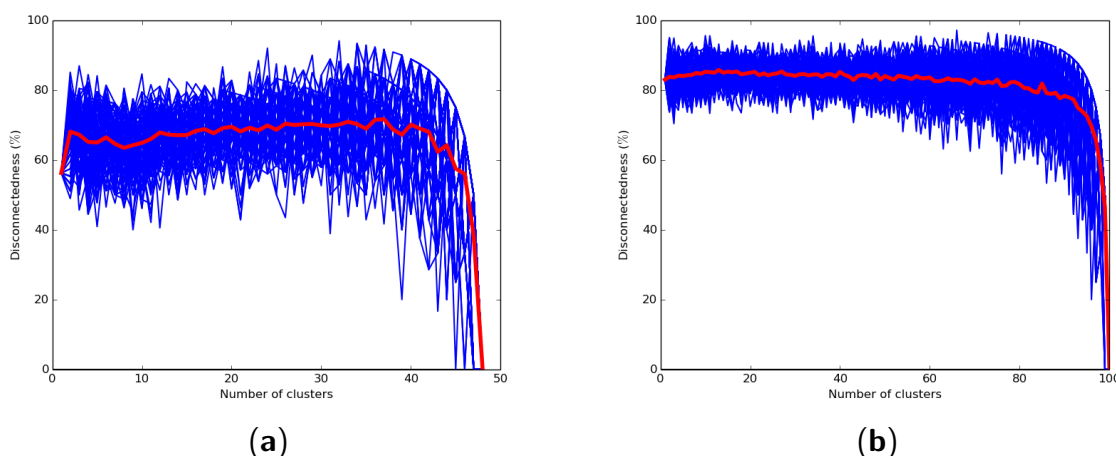


Figure 2.16: K-means algorithm. (a) Average disconnectedness: Cape Town network; (b) average disconnectedness: random network.

On the other hand, Table 2.2 reports on the disconnectedness results of the UAKM algorithm where the

value k was set to four to reflect the optimal number of clusters. For every cluster, the colored and bold nodes in Table 2.2 are the ones that are disconnected from their corresponding cluster heads.

Table 2.2: Average disconnectedness: UAKM algorithm.

Cluster Head	Cluster Members	Disconnectedness
33	38, 47, 37, 35 , 34, 32	66.6
3	25 , 45, 15 , 17, 44, 30, 29, 0, 6, 9	80.0
2	42, 24 , 26, 27, 20, 21, 22, 23, 28, 43, 1, 5 , 4, 7, 8 , 11, 13, 12, 14, 16 , 19, 18, 31	52.2
36	46, 10, 39 , 40, 41	80.0

The results presented in Figure 2.16 for the k -means algorithm show that the expected cluster's disconnectedness was significantly high. They also show that it was close to zero only when all cluster heads were orphan nodes. This could lead to excessive energy consumption resulting from the UAV visiting each and every node of the network for data muling. Figure 2.16b shows that the disconnectedness level of the random network was higher than the Cape Town network disconnectedness. Furthermore, the results presented in Table 2.2 for the UAKM algorithm also show significant disconnectedness in each of the four clusters. This confirms that the k -means algorithms were significantly less reliable than the proposed UADC algorithm.

2.6 Conclusions

In this chapter, a model for optimal sensor network design has been provided where a multi-sink ground-based terrestrial sensor network is expanded by an airborne network of UAVs. The UAVs ferry the sensor data from the sinks of the terrestrial sensor network to the gateway where they would be processed. The coverage problem has been mathematically formulated as an optimization problem aiming at finding the optimal number of clusters to achieve an energy-efficient hybrid terrestrial/airborne sensor network using an UAV as the mobile gateway. A clustering model has been proposed and discussed to address the defined problem. Results of simulations carried out show that the energy spent by the UAV on data muling has the most significant impact on the total energy consumed by the entire data transportation process. The efficiency of the proposed model has been compared with DCC and the k -means algorithms, and the results showed that the proposed mode is more reliable in terms of cluster head selection and overall energy consumption..

3. UAV-aware data routing

The emerging Internet of things (IOT) applications are prompting the deployment of sensor devices in thousands of computing elements into multi-technology and multi-protocol platforms. In this complex system, access to information will be available not only at any time and anywhere, but also using any device. The efficient management of such a complex system is still a subject of research. This chapter considers the case of a ground sensor network assisted by a team of Unmanned Aerial Vehicles (UAVs) to collect sensor readings and deliver them to a base station where they can be processed further. Given a network of ground sensors and the positions of the UAV base stations, firstly a model is proposed for an optimal gateway selection, for efficient data muling by the UAVs. This chapter also revisits the Least Interference Beaconing Algorithm (LIBA) proposed for IOT settings, and proposes preventive mechanisms to ensure that efficient performance in terms of traffic engineering for the emerging IOT. The LIBA based data transport issues are revealed and addressed by proposing a more efficient routing scheme to assist a UAV-integrated data transport system. The problem is mathematically formalised and the underlying data structure is described. The proposed algorithm has been verified to be correct and simulations show that the proposed solutions improve the load balancing and delay handling.

3.1 Introduction

The integration of Radio-Frequency Identification (RFID) and sensor technologies is emerging as an important component of first mile connectivity of the Internet, called the Internet of Things (IoT). Here, information needs to be accessed not only anytime and anywhere, but also by anyone using anything.

A typical IoT deployment scenario consists of a proactive monitoring system, where a network of RFID tags is attached to objects and a set of readers, integrated into sensor motes, is used as an ubiquitous sensor network (USN) [86]. The tags collect information on identification and environmental parameters of the object they are attached to, and transmit this information to a gateway where the information is processed. A variety of services using this information are delivered to users. This concept may be applied in many fields including health-care, environment monitoring and protection, smart cities, public safety and precision agriculture.

The emergence of the IoT has revealed a need for new communication protocols and a redesign of some of the traditional protocols, to achieve efficient routing of information over islands of interconnected lightweight networks. The aim is to support human-to-human, machine-to-machine, and machine-to-human communications. Such protocols require a very high level of survivability, reliability and trust. Failure by these protocols to achieve their assigned tasks may lead to risks that could result in loss of human lives. For example, when a train derailed because the IoT system failed to deliver the right signal to the right node, at the right time, during machine-to-machine interaction. Furthermore, fire-fighters could be misled in a rescue operation during machine-to-human interaction as a result of a faulty IoT visualization algorithm directing them towards an empty room, while the people who need to be rescued are located in another room. Reliability and trust enhancements can be obtained through formal protocol specifications using standard formal methods such as process algebra [87] or Z-notation [50, 88, 51] not only for specification, but also to prove the correctness of some of the fundamental properties of protocols or their underlying algorithms. Formal methods can also be used to discover hidden properties or errors that have never been unearthed. A body of research work on formal verification of network protocols exists but are mostly focused on security related issues.. The correctness of network algorithms and the choice of an appropriate formalism for different types of protocols are two issues which still need to be investigated more by the research community.

The Collection Tree Protocol (*CTP*) described in [89] is a routing protocol which consists of sending periodic routing messages in a network to find paths from each node of the network to the node which initiated the routing. [90] shows that the main *CTP* goals are reliability, robustness, efficiency and hardware independence. Here, simulation shows that this is achieved by using a collection tree and adaptive beaconing features, as described in [89]. However, the algorithm is challenged by the link dynamics and route inconsistencies such as loop creation.

Tiny OS Beaconing (*TOB*) is described in [91] as a protocol with a simplified data structure and hence a simple node structure. This is because in *TOB*, each node's routing table keeps information of only its parent as the next neighbour for the traffic routed from the node to the base station (the sink). This makes the used routing tables simpler than in *CTP*, where nodes keep information about a whole path to the sink. However, *TOB* raises issues including lack of resilience to node failures and also the tree-like many-to-one dissemination model causing uneven power consumption across the network, and additionally, the potential of a big sub-tree being removed from the network in the case of the failure of a single node.

As presented in [92, 93], *LIBP* is the proposed protocol which has *LIBA* as its underlying algorithm. While the *CTP* and *TOB* algorithms may lead to uneven power consumption, the *LIBA* has been proposed as a routing algorithm that uses simplicity to enable scalability of USN. It uses a beaconing process that supports load balancing to improve energy efficiency. Although the *LIBA*, outperforms the *CTP* and *TOB*, its correctness has not yet been formally assessed and it does not consider the length of the path that data need to travel.

On the other hand, weighted centroid localization in Zigbee-based Sensor Networks has been computed in [94]. In order to assist moving agents, a model for decentralized centroid estimation for multi-agent systems in the absence of any common reference frame, has been proposed in [95]. To the best of the authors knowledge, there is no centroid model which considers a ground sensor network and all the possible paths of UAVs moving from base stations.

In this chapter, firstly a model is proposed for determining an optimal centroid taking care of both the ground sensor network and all the possible paths UAVs may take to each sensor, from base stations. Considering the centroid to the sink of the complex network, the *LIBA* issues are pointed out and addressed. In addition the *LISPA* (Least Interference and Shortest Path Algorithm) is proposed, which extends the revised *LIBA* to also minimise the delay in data delivery. The proposed routing algorithm is formally specified and verified using Z notation. Using simulations, the performance of the proposed algorithm is analysed.

The remainder of this chapter is organised as follows. In Section 4.2.3, the problem is defined and the existing best solution is described, pointing out its issues. The proposed solution is presented in Section 3.3. In Section 3.5, the experimental results are provided and Section 3.6 concludes the chapter.

3.2 Observables and problem definition

We use Figure 5.4 to show the considered system. The figure shows a system consisting of a sensor network and three UAVs (U1, U2, U3) and their base stations. It is assumed that a UAV moves from a base station, where it collects data, and moves to a centre where it delivers the data. Each UAV can reach each sensor node using only a specific path.

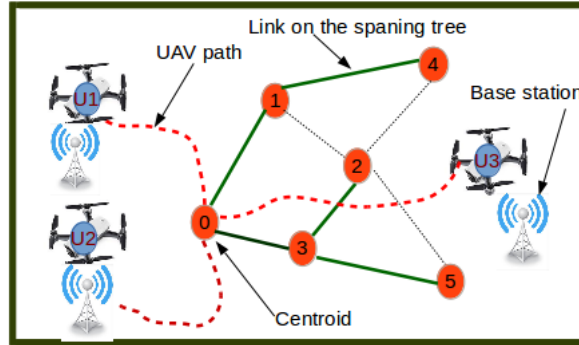


Figure 3.1: The system visualisation.

The problem is defined in two steps.

1. **Gateway finding:** Find a central sensor node which minimizes the average distance to each sensor node and which is best reachable by UAVs. It is necessary to consider the variable Ad_i which expresses the cost based score for the node i in relation to the sensor network. This could be for instance the total/average shortest path from each node to the node i , the total/average delay of data from each node to i or many more. It is also necessary to consider the variable Ap_i to express the UAVs visitation cost. It could be the total/average length of the path from each base station to node i , the total/average delay or rush of UAVs. The centroid problem consists of finding the node i that will minimize the following cost function.

$$Z_i = \alpha Ad_i + (1 - \alpha) Ap_i \quad (3.2.1)$$

where, the parameter α (with $0 \leq \alpha \leq 1$) expresses the variable weights.

This chapter/thesis considers a case where both the variables Ad_i and Ap_i are computable. This is why this problem can be solved in polynomial time by computing the score Z_i for each node the centroid corresponds to and finding the sensor node with smallest score.

2. **Routing to the gateway.** 1. This entails periodically finding the shortest route (in terms of the number of hops and the link lengths) which minimises the interference on the nodes. Here the interference refers to the signal modification in a disruptive manner, as it travels along a channel between its source and receiver. In this thesis, the interference is measure in terms the total number of transferred messages and this is equal to the total number of children a node has been assigned to, in the routing tree.

Here, the routing is done periodically for changing paths so as to share the interference risks. The interference is expressed by considering the number of children in the routing tree, that a node has been supporting , in terms of transmitting the readings to the gateway. This is a dynamic problem which is going to be addressed in the rest of the chapter.

3.2.1 Problem modelling . Firstly, the observable leading to the specification of the assumed network structure, are described using Z notation (see Section 1.4).

Node address

The set of all node identifiers (i.e. addresses) is denoted by IP . The set IP is extended to

$$IP_- = IP \cup \{-\}$$

where, “-” stands for a non-existing or undefined address.

Routing table R

This is a table where a sensor stores information about the network. A routing table of a node contains information about the current path to the sink including the node parent pip , its current set of children cip , its sequence number sn , and the length of its path to the sink hip . The number sn is chosen to be a positive integer showing when last the routing table was last updated, and the number hip is a positive real number showing the lengths of the available paths to the sink. If hip is not yet known for a given node, it takes the value $hip = -$. Hence $hip \in \mathbb{R}_+^+$ where, $\mathbb{R}_+^+ = \mathbb{R}^+ \cup \{-\}$

R
$pip : IP_-$
$cip : \mathbb{P}IP_-$
$sn : \mathbb{N}$
$hip : \mathbb{R}_+^+$
$pip \notin cip$

Note that the notation $pip : IP_-$ expresses the fact that a parent of a node does not necessarily exist.

Updating routing tables: Once a routing tree is changed, each node need to update the information of the routing tree (the routing table).

The ways to update a routing table are as follows:

- **Updating the children lists:** A list of children is updated by making it empty or adding a child to the existing list of children.
- **Updating the sequence number:** The sequence number of a routing table is changed to a new one by replacing an existing number with a new one.
- **Replacing the node's height chh :** The height of a node is the number of fewest hops from that node to the sink of the network. The replacement of a node's height is expressed by Schema chh which shows that all other routing table entries are not changed and the height of the previous routing table $rt?$ is changed to the updated one $h?$.

$chh[R]$
$rt?, r! : R$
$h? : \mathbb{R}_+^+$
$r!.hip = h$
$rt?.pip = r!.pip$
$rt?.cip = r!.cip$
$rt?.sn = r!.sn$


Node N

This is described by its routing table rt , its address ip , its weight wip and its x , y and z coordinates. The weight of a node wip is a natural number which represents the latest number of children of a node.

$N[IP, R]$ $rt : R$ $ip : IP$ $wip : \mathbb{N}$ $x, y, z : \mathbb{R}$
$ip \notin \{rt.pip\} \cup rt.cip$

Network NT

This is determined by its nodes, links and sink. Note here that the weight distribution in the network can be obtained from the nodes distribution, since the descriptors of a node include its weight. A network is described using the following schema:

$NT[N]$ $nodes : \mathbb{P}N$ $links : N \leftrightarrow N$ $nbr : N \rightarrow \mathbb{P}N$ $s : N$ $d : N^2 \rightarrow \mathbb{R}^+$	
$links \in nodes \leftrightarrow nodes$ $ran(link^*) = nodes$ $links^{\sim} = links$ $id_{nodes} \cap linkd = \emptyset$ $nbr(n) = links(\downarrow n \downarrow)$ $s \in nodes$ $s.pip = -$ $\forall n_1, n_2 : nodes \bullet n_1.ip \neq n_2.ip$ $d(n_1, n_2) = \sqrt{(n_1.x - n_2.x)^2 + (n_1.y - n_2.y)^2 + (n_1.z - n_2.z)^2}$ $dom(d) = links$	

As shown in the schema NT , a network of type NT is defined as a connected and bi-directed graph with no self-link. The neighbourhood nbr of a node n consists of nodes connected to n . The sink s is one of the nodes in the network at distance equal to zero. Any two nodes of the network have different identifiers, and the distance between two connected nodes is the Euclidean length of the link connecting the them.

3.2.2 Problem statement. Assume a network G of type NT (see Schema NT), where, periodically, a message is to be sent from or arrive to the network sink $G.s$.

Define L_i to be the length of the path from node i to the sink on a spanning tree T of G .

The problem consists of periodically, finding the shortest and least interfering path with the fewest number of hops, from each node to the sink. That is, find a spanning tree T of the network G which

minimises the following objective function.

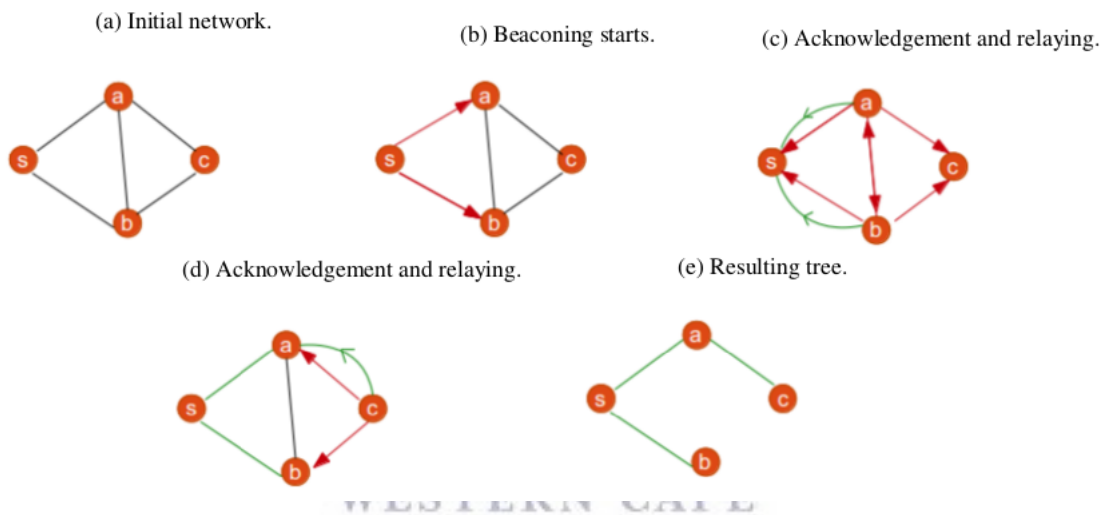
$$\min Z_i = AL_i + (1 - A)(w_i + \#nbr(i)), \forall i \in G.nodes, \quad (3.2.2)$$

subject to $L_i = i.rt.hip$

where, L_i defines the Euclidean length of the path joining node i and the sink s , on T , and $i.rt.hip$ denotes the current height of the node i .

3.2.3 Existing solution and motivation. To the best of the authors, one of the current best approaches for the solution of the problem stated in Section 4.2.3, is the distributed algorithm called Least Interference Beaconsing Algorithm (LIBA). Periodically, it computes the best routing tree from a sink of a network, to minimise interference in the network. Simulations in [93] show that the algorithm outperforms TOB [91] and CTP [90] in terms of energy efficiency. These two algorithms are already implemented in various settings. The LIBA basic protocol is explained using Figure 3.2, which shows a single round.

Figure 3.2: Routing with LIBA.



Consider the initial network in Figure 3.2.a, where node s is the sink. As shown in Figure 3.2.b, the sink starts by broadcasting the beacons in the network.

Figure 3.2.c shows that the neighbours of the sink receive the beacon and relay it in the network, as well as acknowledging the chosen parent in the route from the receiver to the sink. Figure 3.2.d shows that the processes of relaying and acknowledging continue to the remaining nodes with no node acknowledging or relaying a message more than once.

Important note: Each parent receives one acknowledgement message from each of its children if selected by selectors as the minimum weighted parents. The number of received acknowledgement messages becomes the weight of each node. In this example the weight was initialized to zero for all nodes. After the run of one step of LIBA, node a has received one acknowledgement message and hence has a weight equal to 1, whereas node b has weight 0. Since both nodes are connected to node c , in the next LIBA run, node c will choose node b whose weight is least, and the weight of node a will become zero because it will not receive any acknowledgement message, during the routing round.

3.2.4 Data gathering related issues. In this subsection, two major issues for efficient data gathering are discussed.

Node weights update issues

Figures 3.3a, 3.3b and 3.3c are used to present the issues (counter examples).

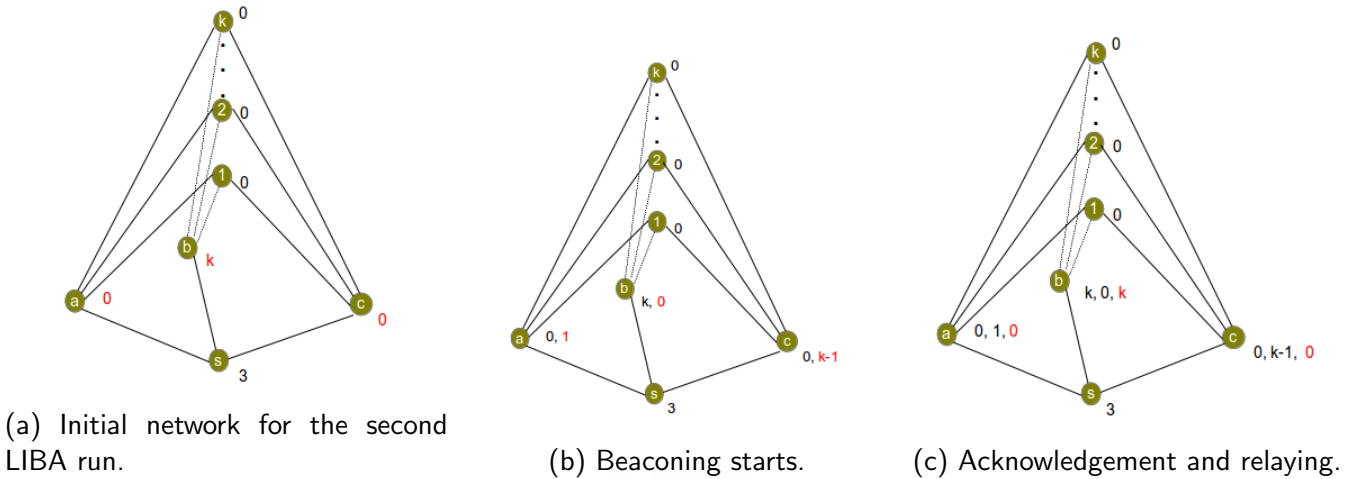


Figure 3.3: Node weights update issues.

Consider the family of three dimensional graphs that are presented in Figure 3.3a, where the set $P = \{1, 2, \dots, k\}$ consists of nodes which are all connected to the nodes a , b and c . The non zero natural number k is chosen to be big for clarifying how effective the solution is, and we assume that the node s is assumed to be the sink of the network. Finally it is assumed that all nodes have initial weight zero (this can be generalized to the case where all nodes have equal weights).

Figure 3.3a shows a case where the sink sends a beacon and, as a result, all nodes in the set P choose the node b and this results in a and c having the weights zero.

In the next LIBA run, the nodes in P have to choose a or c because they have lower weights than b (they both have weight zero). Figure 3.3b shows that only one node from the set P has chosen node a , and all others have chosen node c . At this stage node b has not been chosen and as a result it has the weight zero. Notice that these kinds of choices are possible because in this particular case two candidate nodes have the same weight, and therefore the choice is random.

The problem appears in the third run of LIBA (Figure 3.3c): As node b has a weight less than that of a and c , all nodes in P have to choose node b and its weight becomes k again, whereas nodes a and c update their weights to zero. Notice that the weight redistribution in the third run of LIBA 3.3c is the same as that in the first run (see Figure 3.3a).

Since node a , b and c forward the beacons to the same nodes (in $P \cup \{s\}$), the loads to nodes in P could tend to be balanced if the nodes a , b and c are chosen by around $k/3$ nodes which is the average number of choosers (nodes in P).

Considering the case where initially, all nodes in P chose node b . The weight succession of a , b and c , for many runs is as follows:

- **node a :** $0, w_1, 0, w_2, 0, w_3, \dots$ (the weight of a is either 0 or $w_i > 0$)
- **node b :** $k, 0, k, 0, k, 0, \dots$ (the weight of b is either k or 0)

- **node c :** $0, k - w_1, 0, k - w_2, 0, k - w_3, \dots$ (the weight of c is either 0 or $k - w_i$).

The weight cycling is shown on node a and this shows that the node a is most used at each round with a very high difference, compared with others, in terms of how much it is used. As a result, there is no run at which the load will be balanced differently.

Data transport delay

Figure 3.4 is used to explain this issue. The figure represents a part of a network of type NT where a beaconing message arrives at node s , and it is to be relayed so that node d may use it to choose either node b or c as its parent in the next routing tree.

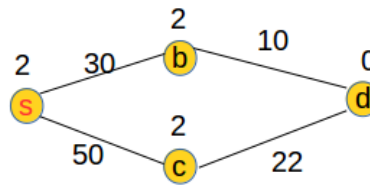


Figure 3.4: data delay issue.

Since both nodes b and c have the same weight (interference), the parent choice for node d is random. However, the choice of node b shows that the data from node d will be transported on a distance of $30+10=40$, whereas with the choice of node c , the distance is 72. This clearly shows that the node c would delay the message from d and hence, in this case, node b is the best choice for moving data from d .

3.3 Proposed solution

The first case under consideration is shown in Figure 3.3a. To ensure efficient load balancing, node a would be prioritised because it has a lower weight (compared with b and c), whereas node b would not be preferred less because it has a greater weight in comparison with other potential parents (a and c). However, since the current algorithm does not prioritise the node a in any way, it is possible to get a case where load balancing fails.

The problem discussed in Section 3.2.4 stems from the fact that the weight computation does not depend on all previously computed weights.

Therefore it is necessary to define a function which takes all previously calculated weights into consideration and returns the weight to a node which prioritises that node node which has supported the least number of children so far. The weight will represent, in this case, the history of the usage of a node and hence its energy consumption.

Consider a case where, a node has to support w_k children at the k^{th} run of LIBA. The following prioritisation function is suggested:

$$W(w_1, w_2, \dots, w_k) = \sum_{i=1}^k w_i \quad (3.3.1)$$

To prioritise a less busy node, we choose to define the function $w(w_1, w_2, \dots)$ is defined considering Figure 3.2.4 it can be seen that at the third run the node a would have a lower total weight (which is

1) and hence a will be the one to be chosen and hence this clearly improves the resource sharing which was the main aim of LIBA.

On the other hand considering the delay issue (as mentioned with Figure 3.4) and to fix it, a node is to be chosen if it is the one corresponding to the shortest path to the sink. The shortest path from the chooser depends on the height of potential parent and also the distance between the chooser and the potential parent.

In this case if a node i is chosen, the length L_n of the path of from chooser n to the sink is

$$L_s = h(i) + d(i, n) \quad (3.3.2)$$

Combining Equations 3.3.1 and 3.3.2, it emerges that on the k^{th} run, the choice of a node i to be the parent of node n depends on

$$Z = AL_s + (1 - A)w(w_1, w_2, \dots, w_k) = A(h(i) + d(i, n)) + (1 - A) \sum_{i=1}^k w_i \quad (3.3.3)$$

where, $0 \leq A \leq 1$ represents the preference constant.

3.3.1 Formalisation of LISPA. LIBA is chosen and prioritisation is applied as shown in Section 3.3 to formalise a more efficient and correct algorithm called the LISPA. To identify the possible observables (used data structure) for the algorithms, it is necessary to refer to the structure of a network, the way messages are sent in the network and the way nodes keep information while the algorithm is processing. Their structures are defined using Z schemas which are translated and explained.

Beacons BC

This is a message broadcasted periodically from a sink, and relayed in a network to enable all nodes of the network to select their (best) parents. A beacon contains the address of its sender sip , the sequence number of the beacon bsn (a natural number which identifies a beacon), the height of the sender sh which helps to determine the nodes ready to be parents (nodes at minimum distances from the sink), the weight sw of the sender, the height of the sender sh , and the x, y and z coordinates of the the sender. fr and the channel used ch . In this particular case the last two variables can be ignored. The set of all beacons is denoted by BC . The schema describing a beacon is as follows:

Acknowledgement message AC

This message sent by a node to only its newly selected parent to confirm the selection. It contains its sender's address sip and the destination dip of the message. The set of all acknowledgement messages is denoted by AC . The following is a schema for an acknowledgement message.

$BC[IP]$ $sip : IP$ $bsn, sw : \mathbb{N}$ $hip : \mathbb{R}^+$
--

$AC[IP]$
$sip, dip : IP$
$sip \neq dip$

The sender of the acknowledgement message is different from the destination of the message ($sip \neq dip$).

Message MSG

In all $PLIBP$ processes, a message is either a beacon or an acknowledgement. Hence the set of all messages MSG are described by the following schema.

MSG
$bc : \mathbb{PBC}$
$ac : \mathbb{PAC}$
$\#ac \leq \#bc$

Since a **beacon** is broadcast and an acknowledgement message is unicast, the number of received beacons is always at least the number of received acknowledgements ($\#ac \leq \#bc$). This condition is also an agreement with the protocol requirement that upon reception of beacons from different potential parents, a node selects only one parent by sending an acknowledgement message to that parent.

Beaconing

In this thesis, the word **beaconing** refers to LIBA routing mechanisms. The beaconing process starts from the sink and goes to all nodes in the same network as the sink. The beaconing processes is described using three Z-classes namely, **Stater**, **Transmitter** and **LISPA**. In this section the classes are studied. This leads to an example showing how the classes cooperate for routing.

Class Starter

This consists of a sink node as its object and the operation $Initiate$ as its method. The sink $sink$ of the network is considered to be constant and it is the same as that in the network nt on which the operation $Initiate$ is used.

The operation consists of initiating the beaconing message $bc!$, and the routing table rt of the sink. In an initiated beaconing message, the sender $bc!.sip$ is the sink's address $sink.ip$, the number of hops $bc!.hop$ is set to zero and the beacon's weight $bc!.sw$ is the weight $sink.wip$ available in the routing table of the sink. The height is set to zero and the x , y and z coordinate of the sender get copied in the beaconing message.

The set $sink.rt.cip$ of children in the sink's routing table is emptied and the sequence number in the routing table is incremented by one.

The initiated beacon has the same sequence number as in the routing table $sink.rt.sn'$.

$nt : NT$
$sink : N$
$sink = nt.s$

Initiate

$\Delta(\text{sink.rt.cip}, \text{sink.rt.sn})$

$bc! : BC$

$bc!.sip = \text{sink.ip}$

$bc!.hop = 0$

$bc!.sw = \text{sink.wip}$

$\text{sink.rt.cip}' = \emptyset$

$\text{sink.rt.sn}' = \text{sink.rt.sn} + 1 = bc!.bsn$

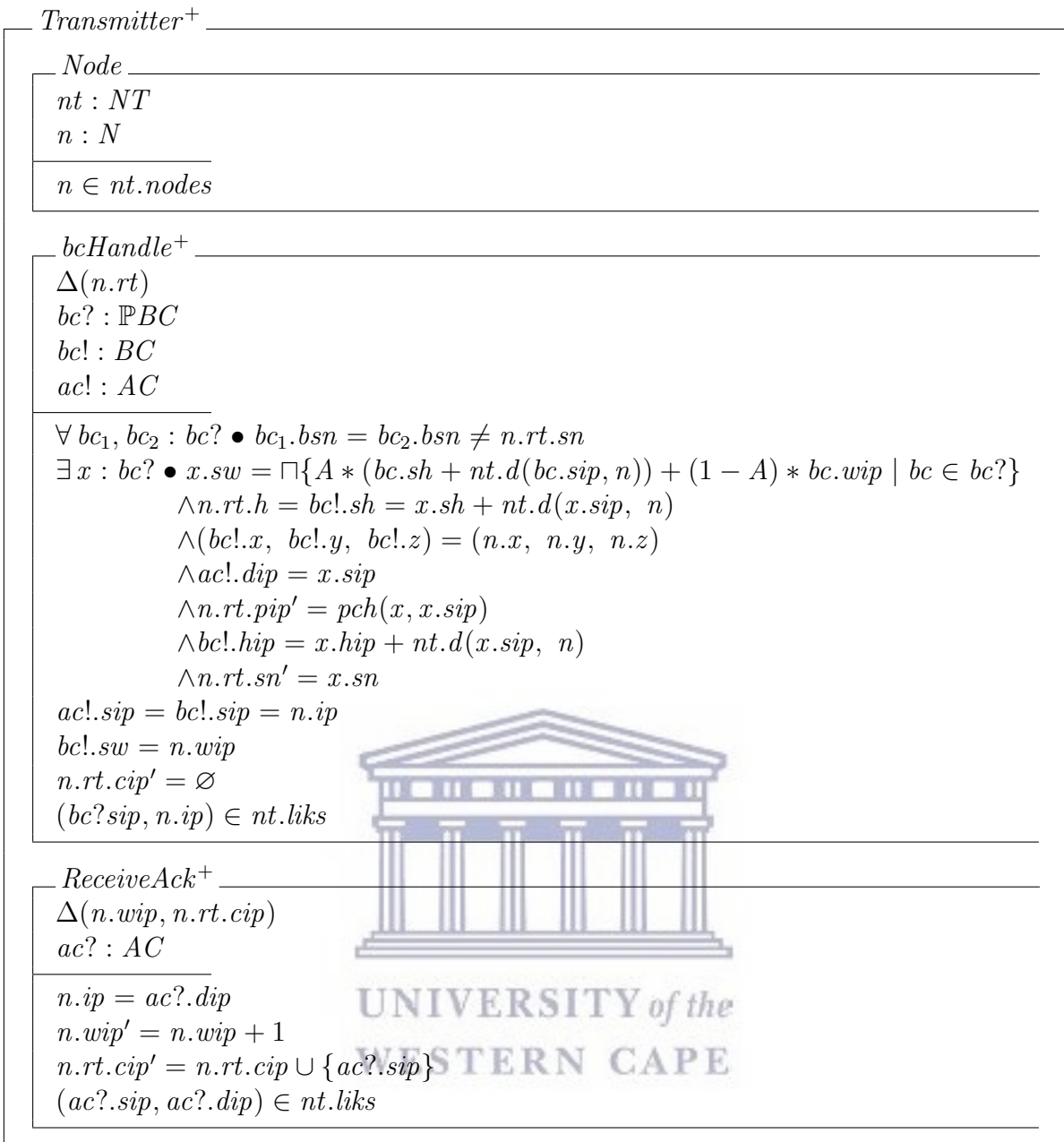
$\text{sink.rt.hip} = 0$

$\text{sink.x} = \text{sink.y} = \text{sink.z} = 0$

Class Transmitter

This consists of one object *Node* and two operations (methods) namely *bcHandle* and *ReceiveAck* as presented in the following schema:





Note: This is the node n of type N in the network nt of type NT .

Operation $bcHandle^+$: This operation takes a set of beaconing messages $bc?$, makes changes to the routing table rt of the receiver of the beacon $bc?$ and outputs the new transformed beacon $bc!$ sends an acknowledgement message $ac!$ and then updates the routing table of the receiver.

To handle the set of beacons requires them to have the same sequence number, and the receiver n handles them if it has a different sequence number. This guarantees the fact that the handled beacons are new.

The node n chooses a beacon x of smallest price z (as calculated in Equation 3.3.3) and uses it to update its routing table and to form the acknowledgement message $ac!$ and the new beacon $bc!$.

A destination in acknowledgement message $ac!$ is a sender of the beacon x and hence the parent $n.rt.pip$ of node n . The height of the beacon $bc!$ is obtained by taking that of the beacon x and

adding the distance between the sender and receiver, while the sequence number $n.rt.sn$ of the node becomes a copy of that of the beacon x .

The sender of the beacon $bc!$ and the acknowledgement message $ac!$ is the node n and the weight in the beacon $bc!$ is the same as the weight $n.wip$ of node n .

The routing table of node n also changes in a way that the set of children $n.cip$ becomes empty.

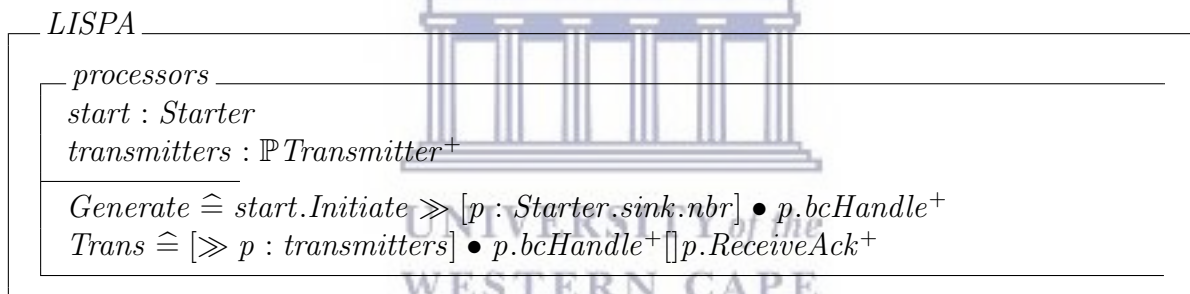
Note that the above changes apply to the node n if the sender of the beacon is connected to it.

Operation $ReceiveAck^+$: is an operation which takes the acknowledgement message $ac?$ and changes the state of its receiver n , provided that the receiver's address $n.ip$ is the same as the destination address $ac?.dip$ specified in $ac!$.

The weight $n.wip$ of n is incremented by one and the sender of $ac?$ becomes one of the children of n . Note here that all this happens if the sender and the destination of the acknowledgement message are connected i.e. $(ac?.sip, ac?.dip) \in nt.links$.

Class *LISPA*

This class consists of processors named *start* and *transmitters* as its objects. The starter *start* is the sink in the network and the transmitters are those nodes in the network which handle the beaconing messages or acknowledgement messages.



The operation *Generate* consists of initiating the beacon which is followed by handling the initiated message by the neighbours of the sink. The operation *Trans* consists of sequentially handling beacons or acknowledgement messages by all concerned nodes in the network.

3.4 Example

The formalism in Section 3.3 is illustrated by an example. The example starts by explaining the notations and colours used and then the LISPA idea is represented using the interpreted graphs.

3.4.1 Notation. In this example, key contents of the schemas are written in tuples for simplicity.

A beacon is represented by $bc(sip, bsn, sw, hip)$ which expresses the beacon bc from the sender sip of sequence number bsn where the sender is at the distance hip from the sink and its weight is sw .

An acknowledgement message is expressed as $ac(sip, dip)$ where sip and dip are the sender and destination of the acknowledgement message respectively.

A routing table looks like (pip, cip, sn, hip) where the node having such routing table has the parent pip , the set of children cip . The last handled beaconing message had the sequence number sn and its sender is at hip units of lengths from the sink.

3.4.2 Colouring logic. In this example the effect of messages is clarified by showing the consequences in the same colours as their causes (messages). After considering an initial network as shown in Figure 3.5a, LIBA⁺ processes are shown by the following colours:

- **Red:** The red arrows show the traces of beaconing messages. The changes in routing tables caused by the beacon, are also shown in Red.
- **Green:** The green bent arrows show the traces of acknowledgement messages, where the green undirected links show the established routes, and the changes in routing tables or weights caused by the corresponding messages are also shown in Green.

Here are two runs of the algorithm, to show all the details.

3.4.3 Steps of a first run of LISPA. Suppose it is necessary to find a path from each node to the sink a in the network shown by Figure 3.5a, where all nodes are assumed to have zero weight.

As shown by Figure 3.5b, the sink a starts by initiating its routing table and broadcasts the beacon $bc(a, 1, 0, 0)$. Its neighbours b , c and d receive it and all update their routing tables. Since each node updates only the beacon from its selected parent (sender whose cost (see Equation 3.3.3) is minimum) and relays it in the network, the beacon from node b is $bc(b, 1, 0, 7)$, as an example.

It is known that a routing table of a node is updated by a beacon from its selected parent. This is why the only difference in the routing tables for nodes b , c , d is the calculated height: they have selected the same parent a . Each node receives a new beacon, sends back an acknowledgement message (green bent arrows) to the selected parent, and relays the beacon from its selected parent.

Figure 3.5c shows the nodes relaying the beacons and acknowledging their selected parents. If a node receives an acknowledgement, it increments its weight by one and then the height of each receiver's height is adjusted by adding the incoming message's height and the distance between the receiver and the sender. This is why the weight of node a has been 3 since it received three acknowledgements messages, whereas the height of node b becomes $7 = 0 + 7$. Note that the node a has rejected the beacons from all its neighbours since they were not new, that is, the sequence number was the same as in its routing table.

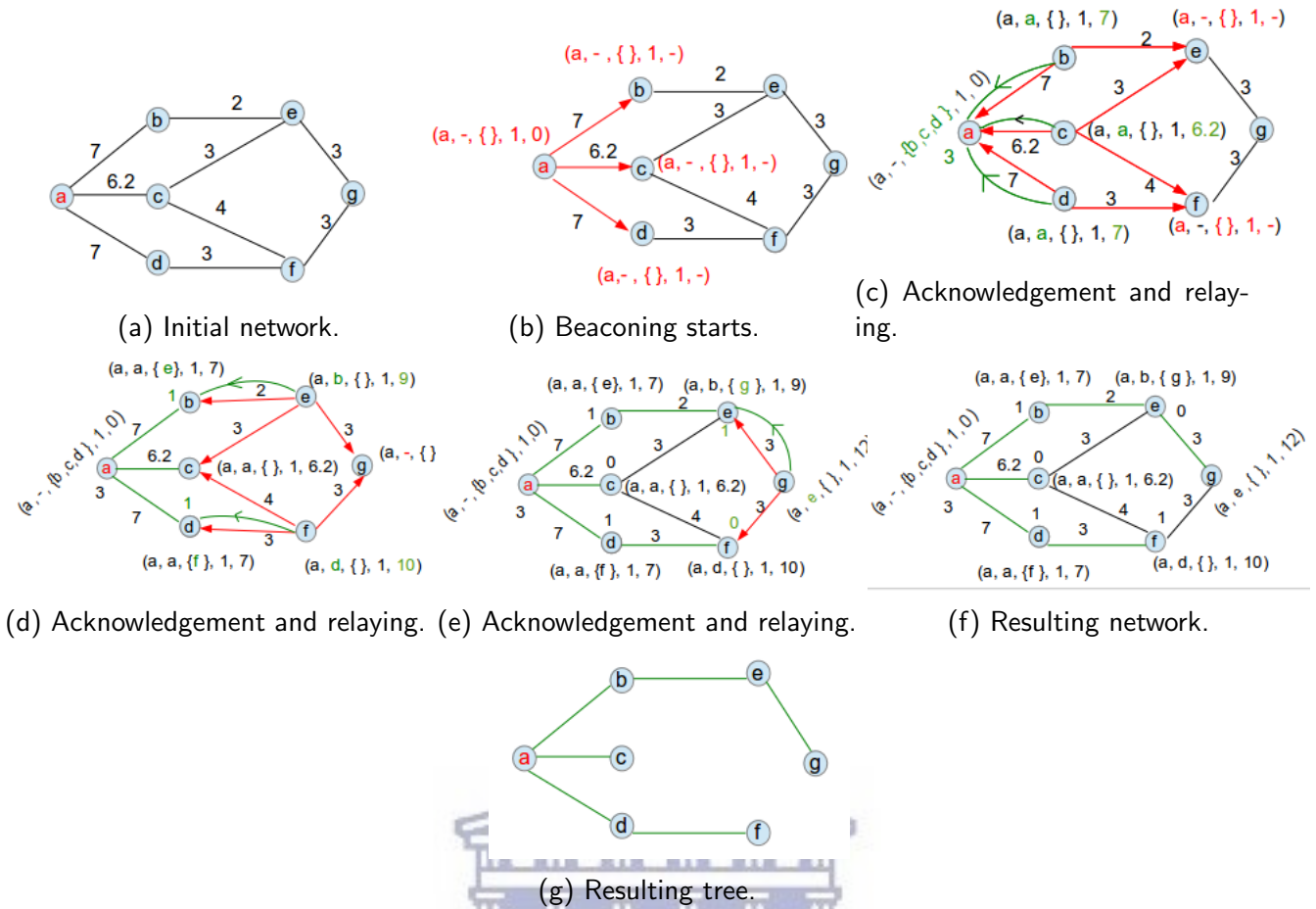


Figure 3.5: First iteration of LISPA.

Figure 3.5d shows a continuation of the process in Figure 3.5c and figure 3.5f shows that the nodes which did not receive any acknowledgement message keep their previous weight (in this case their weights remain zero).

Since each node knows its parent or all its children, the corresponding tree can be extracted as shown by Figure 3.5g by using the trace of the acknowledgement messages.

3.4.4 Steps of a second run of LISPA on the same network. This second run starts with a weighted network with routing tables and the node weights as in Figure 3.5f. The sink *a* starts the process by initialising its routing table and the new beacon.

As shown by Figure 3.6a the initiated routing table of the sink is $r(-, \{\}, 2, 0)$ and consists of no parent, an empty list of children and a new sequence number 2 which is the same as the sequence number of the new beacon originating from the sink *a*. Thus the beacon from *a* is $bc(a, 2, 3, 0)$. Note that the number 2 is obtained by incrementing the previous sequence number by one.

The routing tables are updated by the beacon, by emptying the existing list of children and by replacing the existing parents by a non existing element, and its sequence number becomes a copy of that of the incoming beacon (see 3.6a). This is why, for instance the routing table of node *b*, after acknowledging the beacon from its parent *a*, is $(-, \{\}, 2)$.

Figure 3.6b shows the action of relaying and acknowledging the messages, and the weight of the sink becomes 6 because it has received three new acknowledgement messages. Notice that the height of node *e* changes to 9.2 which does not depends on the previous height but corresponds to the newly established route. Figure 3.6c shows that nodes *e* and *f* make a better choice and choose the node *c* to

be their parent since it has the minimum weight (in comparison with nodes b or d which had also been ready to be parents).

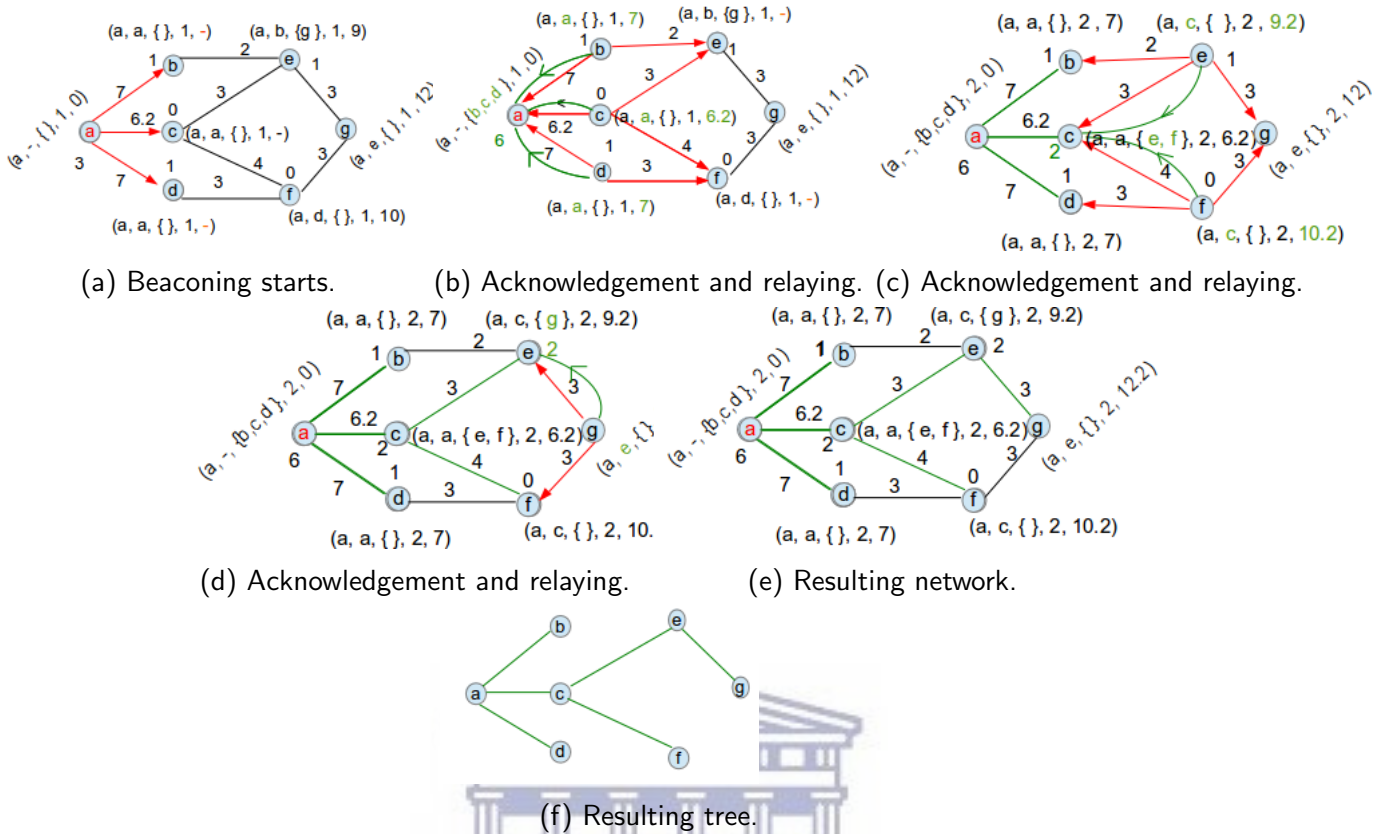


Figure 3.6: Second iteration of LISPA.

Figure 3.6e shows that the nodes keep their weight if they do not receive any acknowledgement message. Based on the new parent selection done so far and the new weight distribution, the resulting tree is shown in Figure 3.6f.

3.4.5 Verification. We use Computation Tree Logic (CTL) is used to express properties and hence verify them.

3.4.6 Theorem. Parent choice

Suppose a beacon is broadcast k rounds by the sink of a network of type NT (see Schema NT in Subsection 3.2.1). At each round, each node that is not the sink keeps on choosing its parent to be its neighbour satisfying the following properties:

- Being closest node to the sink, i.e. a node at fewest hops from the sink.
- Having the minimum weight which means a node of minimal interference.

To express this in CTL syntax, the following predicates must be defined first:

- $beacon(s, k)$: holds whenever the sink s of the network of type NT (see Section 3.2.1) initiates and broadcasts a beacon of type $Initiate.bc!$ (see Schema $Initiate$), for the k^{th} round. That is

$$beacon(s, k) \Leftrightarrow \exists bc : Starter.Initiate.bc! \bullet bc.sip = s \wedge bc.bsn = k.$$

- $n \neq s$: This is satisfied if and only if the node n in the network is different from the sink s of the same network.
- $parent(p, n, k)$: holds whenever node n makes a choice of parent p (its neighbour of minimum weight and nearest to the sink of the network) for its k^{th} round. Note that the number of rounds equals the sequence number of a newest beacon.

Assuming that the predicate $beacon(s, k)$ holds, we proceeding by induction, it can be shown that the CTL formula $AG(n \neq s \Rightarrow parent(p, n, k))$ holds too.

Consider the predicate $bchandle(n, 1)$ which holds if the node n handles the new incoming beacons by the operation $Transmitter^+.bchandle$, for the first round.

From Schema *LIBA* (see operation *LIBA.Generate* and *LIBA.Trans*), it follows that,

$$\forall x : nt.nodes \bullet bchandle(x, 1).$$

On the other hand, $bchandle(x, 1) \Rightarrow (x \neq s \Rightarrow parent(x, 1))$, as shown by SchemaS *LIBA* and *Bchandle*.

Hence,

$$beacon(s, 1) \Rightarrow AG(n \neq s \Rightarrow parent(n, 1)).$$

Assume that $\forall r < k \bullet beacon(s, r) \Rightarrow AG(n \neq s \Rightarrow parent(n, r))$.

It can be verified that $beacon(s, r + 1) \Rightarrow AG(n \neq s \Rightarrow parent(p, n, r + 1))$.

At each node, the r^{th} computed weights can be used to compute the new ones as shown by *Transmitters.ReceiveAc* in Schema *Transmitters*.

In all r beaconing rounds, no operation changes the structure of the network, this is why

$$beacon(s, r + 1) \Rightarrow \forall x : nt.nodes \bullet bchandle(x, r + 1).$$

It should be noted that the sink is the only node which does not satisfy the condition

$\forall bc_1, bc_2 : bc? \bullet bc_1.bsn = bc_2.bsn \neq n.rt.sn$ in Schema *Transmitters*, operation *Bchandle*. Which is why it is the only node in the network which does not choose a parent.

Hence,

$$AG(n \neq s \Rightarrow parent(p, n, r + 1)).$$

3.4.7 Theorem (Correctness of LISPA). Consider the network $nt : NT$ whose sink is the node s . After each run of *LIBA* on nt , the path from each node of the network to s is shortest and consists of the nodes of minimum interference (weight).

The sequence of adjacent nodes and their chosen parents where each node is the parent of the previous one is referred to as **path**.

To express this theorem in *CTL*, the following predicates must be defined.

- $shortest(path(n, s))$: The path $path(n, s)$ from node n to the sink s consists of a minimum number of nodes.
- $cheapest(path(n, s))$: The path $path(n, s)$ consists of a minimum total weight (sum of its nodes weights).

The correctness theorem is then expressed as the following *CTL* expression:

$$AG(\text{shortest}(\text{path}(n, s)) \wedge \text{cheapest}(\text{path}(n, s))).$$

Proof

Suppose $\exists m : \mathbb{N}$ such that at the m^{th} run, the chosen path to the sink is not shortest or cheapest.

Then there exists a node in that path which has a parent not cheapest or closest to the sink. This contradicts theorem 3.4.6.

3.5 Experimental comparison of LIBA and LISPA

In this section, while we study the tendencies of LISPA, we compare it with LIBA using the following performance parameters:

1. **Load balance:** this is measured using standard deviation of the recorded interference as the algorithm runs. It is used to identify which algorithm is better in terms of load balancing in a network. The smaller the standard deviation, the more balanced is the interference in the network.
2. **Average delay:** Since it is assumed that the delay depends on the length of the path taken to deliver messages, it is necessary to study how the delay varies with the runs of LISPA. For every run the average length (on the resulting tree) from each node to the sink is recorded.
3. **Highest cost:** two independent highest costs are evaluated.
 - (a) **The highest interference:** each time the algorithm is run, the highest interfering node is recorded. Since this value increases with more runs of the algorithm, it reflect the life time of the network (the time the highest interference is still less than a threshold).
 - (b) **Longest path length:** the longest path for every run is recorded. This indicates how harmful delays can be, to the underlying network.

3.5.1 The test network. The network under consideration consists of Cape police stations. The network has been extracted from the map as done in Chapter 2, Section 2.5. Figure 2.9 and 2.10b show the map and the corresponding extracted network.

Base station setting

Three base stations have been selected to be in Bellville, Nyanga and Kuilsrivier police stations (nodes 0, 1 and 10).

The sink optimisation

The optimal sink has been identified as a solution to the problem defined in Section 4.2.3, and Equation 3.2.1, with $\alpha = \beta = 1$, has been exploited in the following three steps.

- **Best sink of the ground sensor network:** Here a gateway which is closest to all other nodes in the network has been chosen. This has been done by comparing the single source shortest path to all nodes in the network.
- **Best UAV center:** Given position of UAVs (base stations) the node which is closest to all the base stations has been chosen.
- **Joint best gateway** Considering the two centroids calculated above, the node which is closest to the two has been computed to be node 26, corresponding to Belhar police station.

3.5.2 Standard deviation: load balance. The load balancing property for the two algorithms (LIBA and LISPA) is studied by evaluating the standard deviation of node interference as the run is repeated 3000 times (to address the simulation credibility issues [96]). Since the algorithm periodically sends messages and those messages are the ones which stimulate the increase of interference at each node, the interference standard deviation is plotted against the time which corresponds to the number of runs. Using Python the two algorithms are run by fixing the parameter A (see Equation 3.3.3). The effect of A taking the following values [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1] is also studied. Notice that in the case $A = 0$, Equation 3.3.3 shows that only the Waiting issue as stated in Subsection 3.2.4, and if $A = 1$, we study only the effect of the distance being looked at, as indicated in Subsection 3.2.4.

3.5.3 Standard deviation evolution. Figure 3.7 considers the load balancing as well as the effect of the preference parameter A . The figure also shows that the load balance is best (smallest interference standard deviation), when a focus is put on the waiting (see Subsection 3.2.4). For the LISPA, the figure shows that in the first runs, LIBA is better. However the figure shows that as the number of runs increases the LISPA gets better than LIBA and tends to be at the best point (the point where the interference is 100% preferred, that is when $A = 0$). The figure shows that as the preference in the interference minimisation decreases, the point where the LISPA outperforms LIBA increases (shifts to the right), but in the case where the interference is ignored ($A = 1$), LIBA remains dominant to LISPA.

3.5.4 Delay handling. The delay resolution is observed as the preference parameter is changed. Since it is assumed that the delay depends on the distance traveled by a message to arrive at the sink, the distance is used to approximate the tendency of the delay.

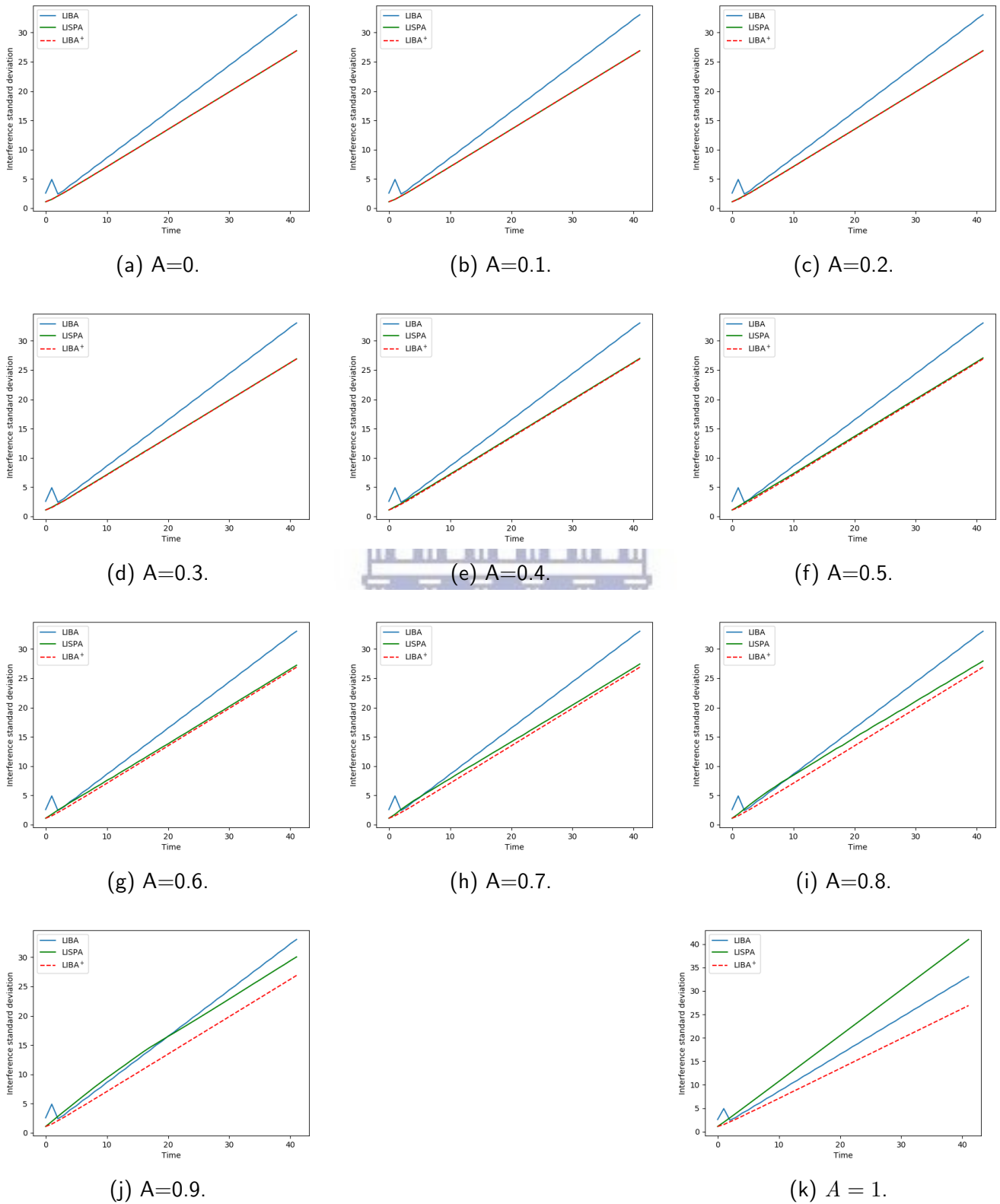


Figure 3.7: Load balancing.

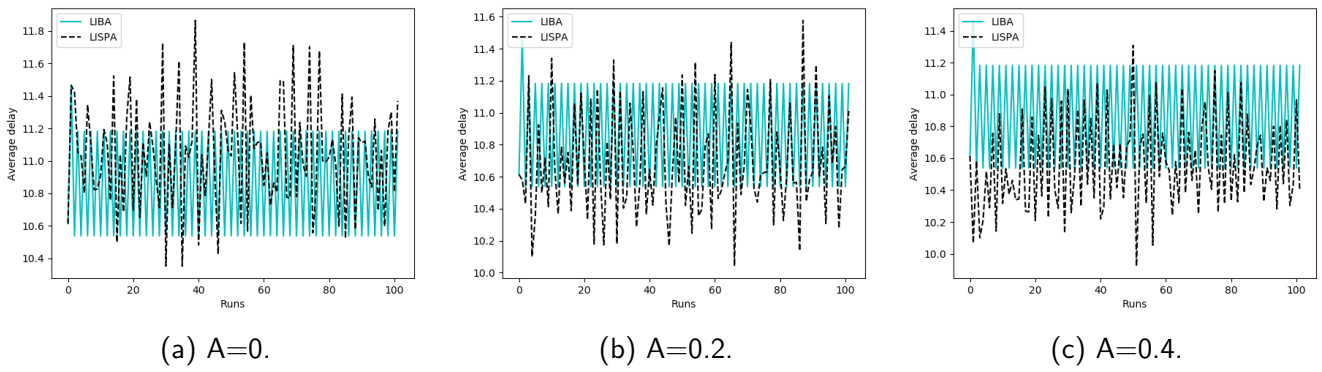


Figure 3.8: Delay evolution.

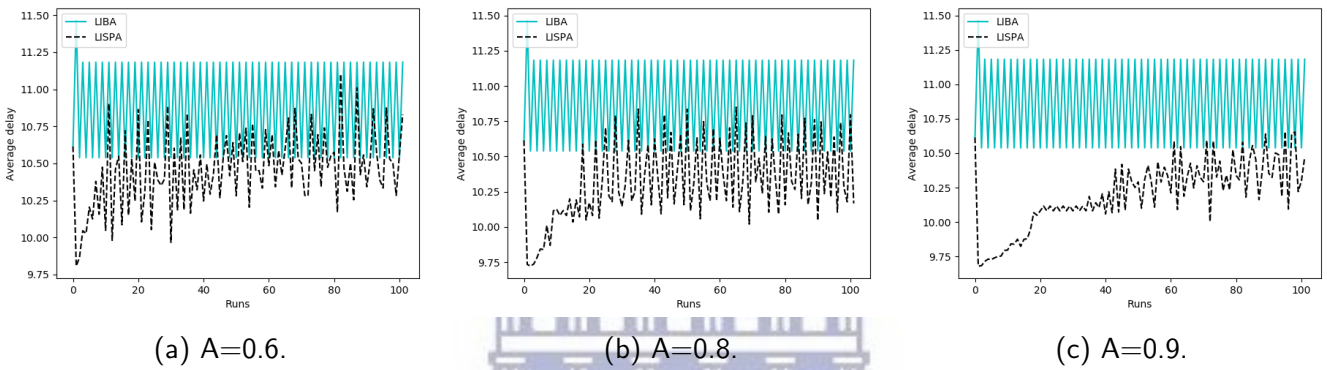


Figure 3.9: Delay evolution.

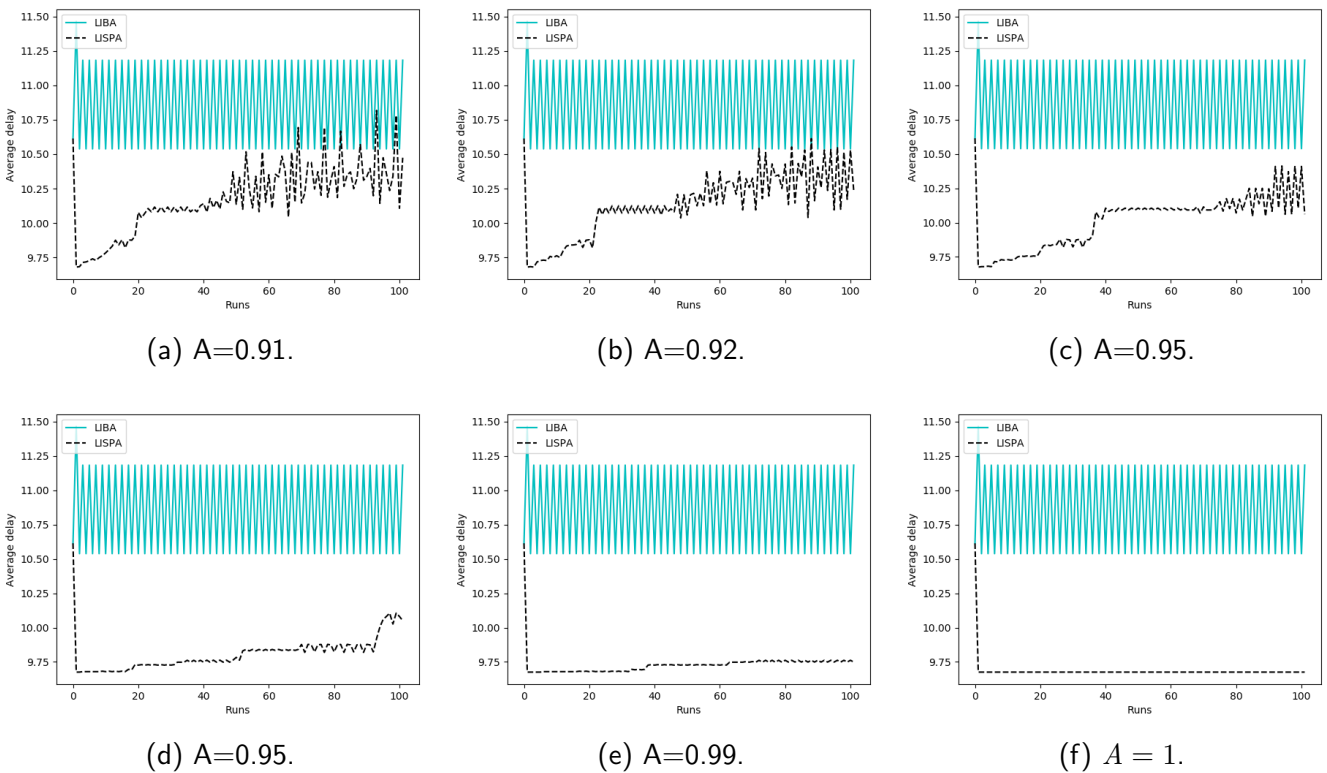


Figure 3.10: Delay evolution.

Figure 3.9 and 3.10 reveal how the delay is improved as LISPA is used and the preference adjust. The figure shows that if the distance is not taken into consideration ($A=0$), the difference between the delay of LISPA and LIBA is not predictable. This explains the fluctuation observed in the figures. From Figure 3.9 up to Figure 3.10, it is shown that as the preference in handling the delay issue increases, the fluctuation and hence the delay decreases. The figure also shows that the delay improvement is best in the first few runs and gets worse as more runs of the algorithm are performed. However, it can be seen that the worst case for LISPA is still better than the case for LIBA.

3.5.5 The highest cost. In this section, 200 runs (iterations) are performed for both LISPA and LIBA on the network shown in Section 3.5.1, to avoid the simulation based credibility issues [96]. The interference is plotted against the names of the algorithms to compare them in terms of the expected highest interference and delay. LIBA and LISPA are compared while the preference parameter is varied.

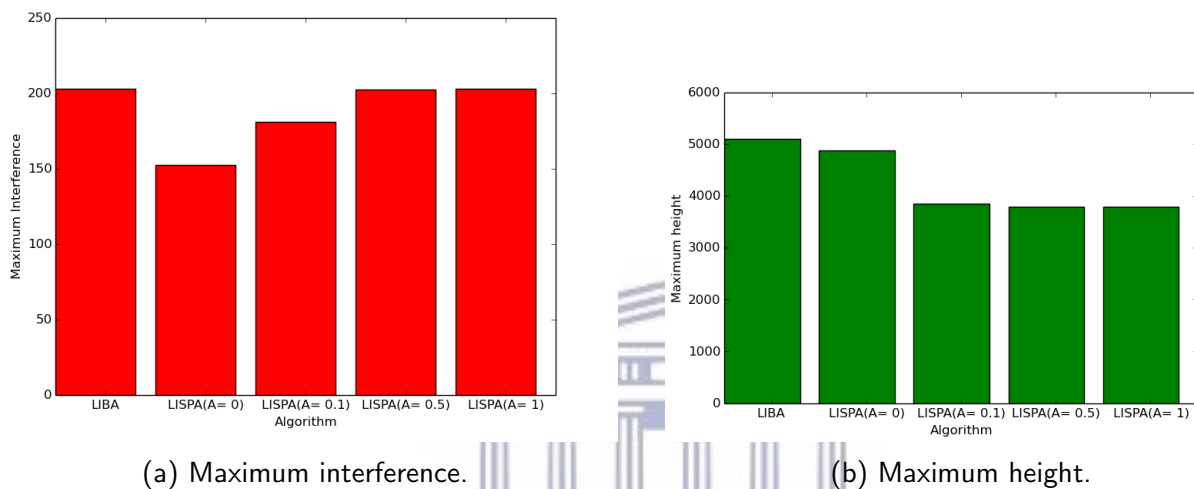


Figure 3.11: Highest cost variation.

Figure 3.11a shows that LIBA has the highest interference. With LISPA, when $A = 0$, the interference minimisation is 100% which results in lowest interference. As the preference on the interference ($1-A$) is reduced for the LISPA, the figure reveals an increase in the interference level towards the corresponding value of LIBA.

Figure 3.11b shows the expected maximum height (length of the path from a node to the sink). The figure reveals that LIBA corresponds to the greatest height and hence the most delaying. This histograms show that as the preference in the delay reduction is increased for LISPA, the maximum height decreases.

3.5.6 Effect of centroid parameter α on the total used energy. The effect of changing the parameter (see α in Equation 3.2.1) changes on the total routing and transport energy is looked at. Figure 3.12a compares the cost corresponding to the optimally chosen centroid, with the average cost when each nodes is elected to be the centroid Figure 3.12b is a zoom in of part of Figure 3.12a to make clear the difference between the two scenarios.

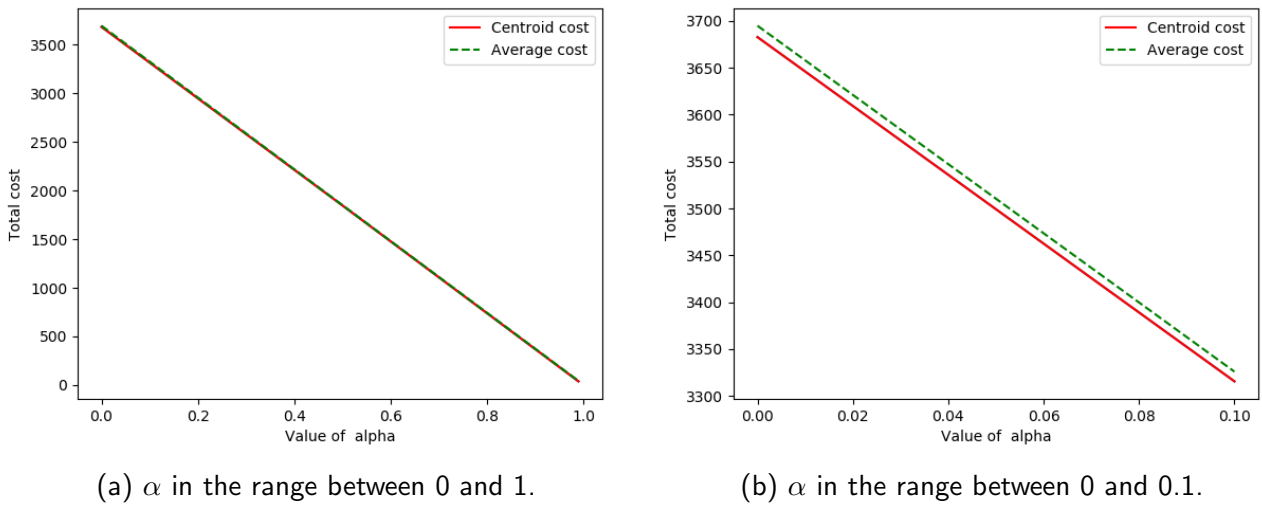


Figure 3.12: Effect of the parameter α .

As shown in Figures 3.12b and 3.12a the total energy for data routing and transport is linearly dependent on the value of α , and it is negatively correlated with the parameter. The graph also reveals that that the optimal centroid (the one corresponding with the lowest cost) always exists.

3.5.7 Impact of α on the centroid choice . If the value of α is changes, it is interesting to see the resultant changes in the centroid election.

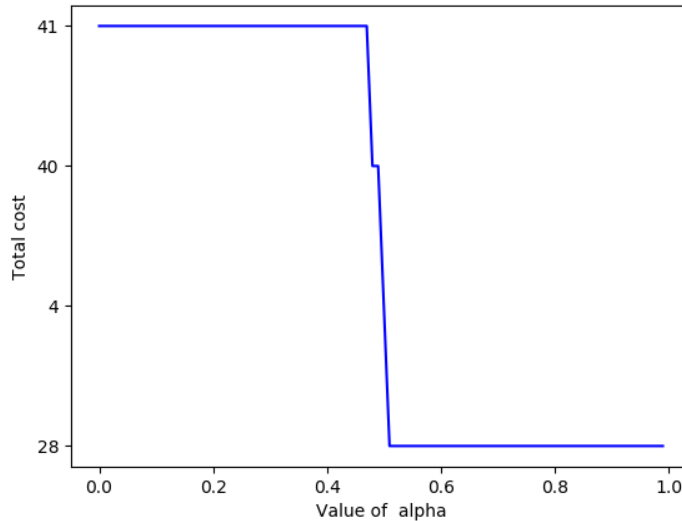


Figure 3.13: Centroid variation.

Figure 3.13 shows that when $0 \leq \alpha < 0.48$, then centroid is constantly 41. When $0.48 \leq \alpha < 0.5$ the optimal centroid is 40 and the rest of interval up to 1 the optimal centroid is 28. Note here that nodes 40 and 41 are the most elected to be centroid.

3.6 Conclusion

In this chapter, the UAVs-aware data gathering model was presented. It is a two steps model, with steps summarized as follows: at the first stage, a UAV sensor aware gateway is optimally selected; while at the second stage the Least Interference and Shortest Path Algorithm was proposed to minimise the interference and the delay experienced by messages in the network. Z notation has been used to formalise the problem and propose the algorithm. The proposed algorithm was verified and proven to be correct. The performance analysis has been done using Python simulations and obtained results show that the LISPA outperforms the List Interference Beaconing Algorithm (LIBA), in terms of delay and load balancing in the network.



4. Cooperative Model for one Target Visitation

This chapter considers a case where a team of Unmanned Aerial Vehicles (UAVs) is to successively and persistently fly over a given area of interest, coming from different locations with the mission of collecting sensor readings from these locations and delivering/muling these readings to a fixed location, where a relevant data infrastructure is available for processing the readings, and the related information is provided and/or shared as a service to the community. A scenario is considered where data delivery guarantee, at the processing location for a period of time called revisit deadline, is a key feature upon which a successful mission depends. The problem of satisfying the revisit constraints for successive delivery is mathematically formulated and justified to be intractable. Furthermore, the collision issues are addressed. Three heuristics to solve the problem are suggested and compared through numerical simulations. Lastly, the effectiveness of both heuristics is analysed and verified.

4.1 Introduction

Wireless sensor networks have been extensively used as the rough material of the emerging Internet-of- Things (IoT) for developing large scale monitoring applications in harsh environments. In some situations, where sensors need be deployed over large areas, it can be very constraining to provide a continuous network connection to a sink node for each sensor node. The reasons for this are numerous and varied. First, for the sake of cost reduction, there is no need to deploy sensors in the entire area of the network. Second, the geographical characteristics of some area (rivers, mountainous area) could make the deployment of physical sensors impossible. Third, in the case where sensors are attached to mobile/nomadic entities, the deployment of physical relays and sink nodes everywhere could become tedious. Fourth, if a natural disaster should occur, the network could suddenly be transformed into a set of isolated islands of nodes that would not be able to provide the communication to relay urgent data or to summon the required immediate assistance.

Faced with the deployment constraints described above, the use of unmanned aircraft vehicles, also often referred to as drones, could become an alternative solution, replacing fixed sink nodes and reducing the number of sensors that need to be deployed. In this context, one can imagine an Internet-of-Things in motion scenario where a group of ground sensors deployed in different geographic locations are being successively and persistently visited by a team of UAVs to collect their sensor readings and transport/ferry these readings to a given target location such as a data centre with relevant capabilities for processing data and sharing the resulting information as a service to the community. Such a target has to be revisited after a given period of time referred to as the revisit deadline. When two UAVs are close to each other, communication between them should prevent a possible collision.

Many emerging application scenarios are revealing the relevance and benefits of using UAVs/drones in “Internet-of-Things in Motion” settings for public safety in the developing world. Furthermore, when deployed in environments with ground-based sensors and flexible gateways [97], an IoT-in-Motion network has the potential to replace long distance sensor deployments [98] as they can play the role of mobile sinks in regions where a mesh deployment is not an option, such as in hilly or mountainous regions where wireless communication is an issue.

4.1.1 Related Work. Scheduling the UAV visitation has initiated several avenues of research. Motivated by the scenario of persistent visitation with fuel constraints in [99], a model for persistent visitation under

revisit deadline constraints has been proposed in [100]. In this case, a single moving agent is to visit several targets optimally, each of which has the revisit deadline. The models from these two pieces of research have been considered in [34] to suggest a cooperative model to visit many targets using a team of UAVs, for surveillance and pursuit purposes. In this study the UAVs do not communicate with each other. They rather rely on the information from the ground sensors optimally placed as suggested in [35], to know their future actions. The problem for multiple UAVs to visit a single target has been addressed in [11], where the target (area) is optimally partitioned using straight lines, and the visitation assignment is done using an auction mechanism. This approach would benefit the target which requires the simultaneous visitation of UAVs.

In [101], sweeping and monitoring a changing environment using multiple robots has been addressed. The presented model assumes that the robots cannot communicate with each other, and that each UAV is assigned the pre-computed task to be performed with a controlled speed. In [102], each point of the area is visited on a maximal frequency basis. The uniform frequency is maximised. This is achieved by introducing the spanning tree patrolling approach, which consists of finding the Hamiltonian paths to be used by robots. Based on the paths, task assignment follows and satisfies the time optimality. Frequency-based multi-robot patrolling has been addressed in [103], where the visitation frequency was optimised. On a cooperation basis, robots are assumed to have uncertain velocities, and move back and forth along a polygon such that the assigned areas of movement overlap. In this case, robots visit different areas which do not have fixed revisit deadlines.

Frequency-based patrolling has been applied in waste cleaning and monitoring (see [104, 105]). Continuous sweeping based on robots cooperation has been addressed in [106], where the sweeping frequency was not necessarily uniform. The main focus here has been to achieve the adaptive area partitioning of a dynamic environment for assigning tasks to robots. In [107], many targets are visited by many robots with the goal of maximising uniform frequency (each target is covered in the same optimal frequency).

In [108], a UAVs coalition model has been suggested for a single target search and prosecute mission. The model intends to minimise the target search time which is not feasible for assistance related visitations, where the visitation (assistance) time needs to be maximised.

More transportation problems have been addressed in [109], but none of them has addressed the one suggested here in this thesis, i.e, a periodic visit of one site with a fixed deadline.

4.1.2 Contribution and Outline. This chapter presents a persistent scheduling model to enable a team of UAVs to visit a target, optimising a multi-objective UAV team's mission. Two objectives are considered: a) maximization of total time taken by all UAVs to arrive at the target and b) minimization of the overdue time to visit the target subject to the revisit deadline and environmental constraints.

The contributions are two-fold. First, the problem of finding the optimal schedule to send different UAVs (which have different properties) to visit one location in the network is mathematically formalised. Such a schedule aims to minimize the missing of deadlines and to maximize the time between the first and last visit (here, we need to use the team of UAVs to assist the target and this needs to be done as long as possible). Second, the author is able to prove that the problem is NP hard, and hence it can be addressed by proposing three heuristic solutions, which are compared using simulation. The effectiveness of all three heuristics is proven to ensure the UAVs persistent and efficient visitations.

The remaining part of this chapter is organized as follows. Section 4.2 presents the cooperative data muling model by sketching a deployment scenario, formalizing the problem and describing its parameters. In Section 4.3, heuristic solutions to the problem are presented and explained. In Section 4.5, a simulation is conducted to compare the heuristics, and an analysis is conducted to prove the persistent effectiveness of these heuristics. Section 4.8 concludes the chapter.

4.2 The Cooperative Data Muling Model

In this section, first a deployment scenario is presented and thereafter the visitation problem is described together with its corresponding parameters. Finally, the problem is mathematically formulated as an optimisation problem.

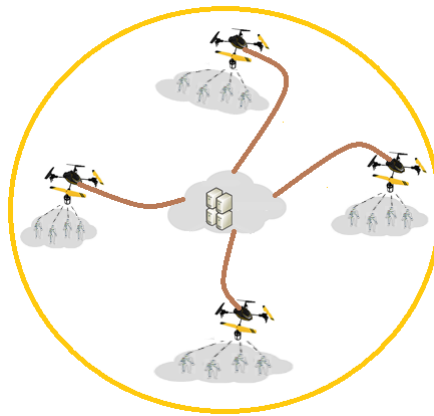


Figure 4.1: Visualisation of UAVs on action.

4.2.1 Deployment Scenario. Figure 4.1 depicts an application scenario in which the use of UAVs can be synonymous to mobile sink nodes or to mobile resources carriers. In such a scenario, a number of public health persons carrying wearable body sensors are tasked to collect health, environmental, chemical, and biological data in a region affected by epidemics such as Ebola or Malaria. In the case where sensitive data are discovered by the staff, their delivery to the control centre should be done urgently. Moreover, an immediate physical assistance (e.g. physical rescuing) might be required in emergency situations. As the number of UAVs and their available resources (such as battery energy) to assist at the destination are assumed to be limited, the visiting (assistance) time has to be maximized, under the constraint to periodically arrive at the destination by the deadline. However, given the UAVs placement (with respect to the target), the trajectories and the states of UAVs, it might be impossible to satisfy this condition. This is why an optimal visitation schedule should be designed to minimize the overdue time to visit the target and to maximize the duration of the assistance.

It is assumed that the UAVs are able to cope with the environment by adjusting their speed to keep their flight time unchanged.

4.2.2 Problem Formulation. While moving, each UAV, say i , is assumed to travel at a speed v_i in a three dimensional space. There is a single place to be visited and each UAV i comes from its position (x_i, y_i, z_i) and travels in its own three dimensional trajectory expressed as $R^i(t) = (R_1^i(t), R_2^i(t), R_3^i(t))$, where $R_1^i(t)$, $R_2^i(t)$ and $R_3^i(t)$ respectively represent the x, y, z position of the UAV i at time t .

UAVs may handle the collision avoidance themselves by changing their flying routes to different altitudes. This is only done when they are on a collision course and are close enough to communicate with each other.

We first use Z notation to specify the type of UAVs, target, visitation process, and the initial conditions.

The type U of UAVs.

The state of each UAV is denoted by Schema U and is determined by the position function of time \mathcal{R} , the time period t of its flight, its source s , its destination d and its precision factor $alpha$ which expresses

the environmental and physical properties of the UAV (it is assumed that each UAV knows about its path and its own properties, otherwise α would be computed using machine learning techniques).

U
$\mathcal{R} : \mathbb{R}^+ \rightarrow \mathbb{R}^3$ $t : \mathbb{R}^+$ $s, d : \{(0, 0, 0), \mathcal{R}(0)\}$ $\alpha : [0, 1]$
$s \neq d$ $t = 0 \Rightarrow d = (0, 0, 0) \wedge s = \mathcal{R}(0)$

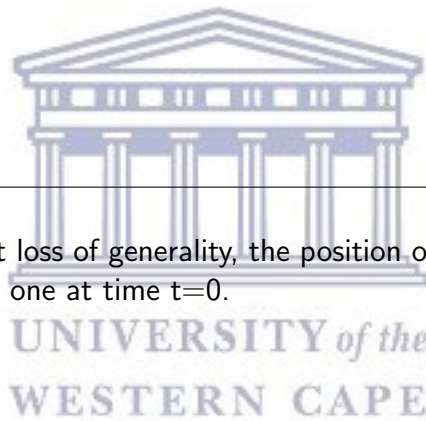
The source and the destination are always different, and initially ($t=0$), the destination is the origin (assumed position of the target) and the source is the initial position of the UAV.

The type T of the target.

The target T is determined by its position p and its revisit deadline r (r is the time window between any two visit).

T
$p : \mathbb{R}^3$ $r : \mathbb{R}^+ \rightarrow \mathbb{R}$
$p = (0, 0, 0)$ $\forall t \in \mathbb{R}^+ \bullet r(t) \leq r(0)$

As shown in Schema T and without loss of generality, the position of the target is the origin in \mathbb{R}^3 , and the maximum revisit deadline is the one at time $t=0$.



The visitation $Visit$.

The target visitation is done by one UAV at a time. Once the UAV reaches the destination, the revisit deadline is reset to its initial value (maximal value). Keeping the same speed, the UAV reuses its former trajectory to go back to its source. This is formally expressed by the Schema $Visit$.

Visit[T]
$\Delta T.r$ $u!, u? : U$
$u!.t > u?.t$ $u!.s = u?.d$ $u!.d = u?.s$ $u!.R = u?.R$ $T.r' = r(0)$

Initial condition

Each UAV starts at its station (which is different from the target), and the revisit deadline is set to its maximal value.

$\text{Init}[U, T]$ $\mu : \mathbb{P}U$ $\tau : T$
$\forall u \in \mu \bullet u.d = (0, 0, 0) \wedge u.v = 0$ $\tau.r : \{0\} \rightarrow \mathbb{R}$

The UAVs and their paths have different physical properties and hence have different values of α (here α satisfies the condition $0 < \alpha_j \leq 1$ and expresses the precision depending on the environment and the physical conditions of underlying UAV).

Time modelling

Let t_i^d be the departure time of the UAV i to the target and let t_i^a be its arrival time. In this case, the time t_i taken by the UAV i to arrive at the target is

$$t_i = t_i^a - t_i^d \quad (4.2.1)$$

The arrival time of the next UAV say j abides by the deadline r rule. This is done by allowing j to arrive after r units of time after the UAV i has arrived. That is

$$t_j^a = t_i^a + \alpha_j r, \quad (4.2.2)$$

It follows from Equations 4.2.1 and 4.2.2, that the departure time of the next UAV j is computed as follows.

$$t_j^d = t_i^d + \alpha_j r + t_i - t_j. \quad (4.2.3)$$

The equation 4.2.3 is only valid if $\alpha_j r + t_i \geq t_j$ and otherwise $t_j^d = t_i^d$ since t_i^d is considered to be the initial time for the UAV j .

Notice that given that an UAV U_1 is to travel before U_2 at time t_1^d . The departure time of U_2 can be deduced from t_1^d but for this work the flight time of the next UAV is computed using the first order Markov process. That is, the next travel time only depends on the previous one.

4.2.3 The Optimisation Problem. We assume a team U of UAVs and let I be the set of the first $\#U$ indices. The problem consists of, for each UAV n , finding the departure time to the destination to minimise the missing of the revisit deadline and to maximise the time between the first and last visit. This consists of finding the bijection $b : U \mapsto I$ to minimize the cost

$$C_b(T) = \frac{\beta}{\gamma} \times \frac{t_1^a - \alpha_1 r + \sum_{n=2}^{\#U} [t_n^a - t_{n-1}^a - \alpha_n r] u(t_n^a - t_{n-1}^a - \alpha_n r)}{\sum_{n=2}^{\#U} (t_n^a - t_{n-1}^a)}, \quad (4.2.4)$$

where, u is the unit step function, which is equal to one in the case where the arrival of the next UAV is stale. That is

The numerator represents the total overdue time corresponding to the bijection (schedule), whereas the denominator represents the corresponding total time to visit the target. The fraction $\frac{\beta}{\gamma} > 0$ represents the optimisation preference. That is

- $\frac{\beta}{\gamma} < 1$: Emphasis is placed on the total used time maximisation
- $\frac{\beta}{\gamma} > 1$: Emphasis is placed on minimising the missing of the deadline.

We aim to reduce the problem 4.2.3 to the Dynamic Assignment Problem (see Section 1.3.2), which is an NP-hard problem (see [47]). Consider the list U of UAVs to be the resources, and the set I (with $\#I = \#U$) set of ordered indices which is considered to be the set of tasks. In this case given that the UAV U_i is assigned to the index $k \in I$ it will be the k^{th} to depart to the destination.

Observe that the cost of making the $(k + 1)^{\text{th}}$ depends on the $(k)^{\text{th}}$ assignment (see Equations 4.2.3 and 4.2.4). The dynamism of the assignment cost qualifies the problem to be a case of the Dynamic Assignment Problem, and the reduction is clearly linear. This makes us conclude that the problem 4.2.3 is NP-hard.

4.3 Heuristic solutions

In this section, three heuristic solutions are presented in the form of algorithms.

4.3.1 The First Deployment algorithm (FD). This algorithm consists of iteratively finding the next best UAV to depart. The best UAV to depart is the one which minimises the slack and overdue time to arrive at the destination. Note here that the overdue time is the positive difference between the waiting time of the destination and the revisit deadline, whereas the slack time has a negative value.

Selection of the first UAV to go

Input:

- ↔ A hash table whose list of keys is U and contains the UAVs labels, where its list of values is T and consists of the corresponding time to the target (destination).
- ↔ The revisit deadline for the destination r .
- ↔ The list α containing each UAV's accuracy.

Output:

- ↔ The UAV to go first.

Algorithm 4: $First(U, T, r, \alpha)$.

```

1  $D = \{T_i - r \cdot \alpha_i \mid T_i \in T\}$ 
2 if  $D_i \geq 0 \forall i \in U$  then
3   | return  $i, T_i$  and  $\alpha_i$ , with  $D_i = \min\{D\}$ 
4 end
5 else
6   | return  $i, T_i$  and  $\alpha_i$ , with  $D_i = \max\{x \mid x \in D \wedge x < 0\}$ 
7 end

```

The algorithm starts by calculating the set D of the overdue (positive values) and slack (negative values) times corresponding to the deadline r . As shown in Algorithm 4 the first to go is obtained by selecting the UAV which is closest to the revisit deadline.

Note that Line 2 shows that the time complexity for the the algorithm is $\mathcal{O}(\mathcal{U})$, where \mathcal{U} represent the size of the set U .

The next deployments computing

Input:

U , T and r as explained in Algorithm 4

Output:

A set U_d of UAVs and their departure time set T_d .

Algorithm 5: Departure1(U, T, r).

```

1  $u \leftarrow First(U, T, r).i$ 
2  $a_u \leftarrow First(U, T, r).\alpha_i$ 
3  $t_u \leftarrow First(U, T, r).T_i$ 
4 The departure time of the first UAV  $t_1^1 \leftarrow 0$ 
5 List of UAVs ordered in terms of departure time.  $U_d \leftarrow [u]$ 
6 Initial departure time  $T_d \leftarrow [t_u]$ 
7 Schema Init
8  $U \leftarrow U \setminus \{u\}$ 
9  $\alpha \leftarrow U \setminus \{a\}$ 
10  $T \leftarrow T \setminus \{t\}$ 
11 while  $U \neq \emptyset$  do
12    $T_s \leftarrow \{|T_i - t_u|, \forall T_i \in T\}$ 
13    $u \leftarrow First(U, T_s, r).i$ 
14    $a_u \leftarrow First(U, T, r).\alpha_i$ 
15    $t_u \leftarrow First(U, T_s, r).T_i$ 
16    $U \leftarrow U \setminus \{u\}$ 
17    $\alpha \leftarrow U \setminus \{a_u\}$ 
18    $T \leftarrow T \setminus \{t_u\}$ 
19   Append  $u$  in  $U_d$ 
20   if  $r \cdot a_u + t_1 - t_u \geq 0$  then
21      $t_1^1 = t_1^1 + r \cdot a_u + t_1 - t_u$ 
22   end
23   Append  $t_1^1$  in  $T_d$ 
24    $t_1 = t_u$ 
25 end
26 Return  $U_d, T_d$ 

```



The algorithm first computes the first UAV to travel, i.e. its identity u , its accuracy, a and its corresponding time to the destination t_u and its departure time is set to 0. U_d and T_d are then updated by appending u and t_u respectively. At Line 6, Schema *Init* is used to initialise the system. The chosen UAV is then removed from U , the corresponding accuracy gets removed from the list α and the corresponding time is removed from T (Lines 7- 9). A set T_s of time differences to t_u is computed. The update of U , T , U_d and T_d is done until U gets empty. In each update case, if the deadline is not overdue, the flight time is set to the time closest to the revisit deadline. Otherwise, the departure time of the current UAV

is set to the preceding one (Lines 16 and 17). Furthermore, since the time complexity executing Lines like Lines 13 or 14 is $\mathcal{O}(\mathcal{U})$ and this is computed on a loop of whose longest size is \mathcal{U} , it is clear that the time complexity of this algorithm is $\mathcal{O}(\mathcal{U}^2)$.

Algorithm 6: $\text{Departure2}(U, T, r, \alpha)$.

```

1 Make a dictionary  $D$  sorted with respect to the time each UAV takes to arrive at the destination
2 Let  $V$  and  $K$  be the values and the keys for the dictionary, respectively.
3 Initialise the departure time of the first UAV at  $t_1^1 = 0$ 
4  $Dd_{K_0} = 0$ .
5 for  $i \in \{1, 2, 3, \dots, |K|\}$  do
6   if  $r \cdot \alpha \cdot K_i + D_{K_{i-1}} - D_{K_i} \geq 0$  then
7      $Dd_{K_i} = Dd_{K_{i-1}} + r \cdot \alpha \cdot K_i + D_{K_{i-1}} - D_{K_i}$ 
8   end
9   else
10     $Dd_{K_i} = Dd_{K_{i-1}}$ 
11  end
12 end
13 Return  $Dd$ 

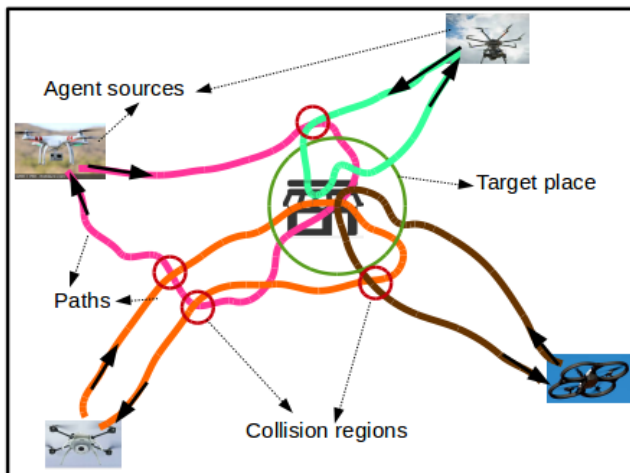
```

4.3.2 The first to arrive, the first to go algorithm (FAFG). As shown in Algorithm 6, the flight time of the first UAV is initialised to zero. The departure time for each UAV is computed with respect to its preceding UAV in the sorted dictionary D . Lines 6 and 7 show that in the case where waiting can not cause the UAV to be late to arrive to the destination, it has to wait for a maximum time. On the other hand Line 9 shows that in the case where the waiting would cause the violation of the revisit deadline policy, it has to depart at the same time as the preceding UAV. Note that the complexity of the Algorithm 6 is $\mathcal{O}(\mathcal{U})$ since the size of k is the same as the total number of UAVs.

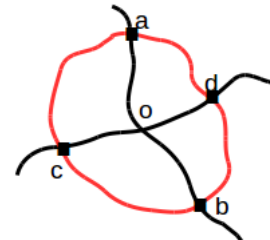
4.3.3 The closest to the expected arrival time of UAVs, the first to go (CFG). This algorithm differs from Algorithm 6 only at Line 1. The order is done with respect to the difference between the arrival time for each UAV and the average arrival time.

4.4 Collision recovery

In this section, collision issues are addressed. It is assumed that collision points are known and that the time for each corresponding UAV to arrive at that point is known.



(a) Paths visualisations.



(b) A collision region.

Figure 4.2: Collision visualisation.

Figure 4.2a shows a view of the UAVs paths where possible collisions are indicated by circles. Figure 4.2b shows a zoom in of the collision region where, the time taken for travelling along sub-paths AO, OB, CO and OD is the same as ϵ .

Algorithm 7: Is_Colliding(A,B).

```

1  $I \leftarrow$  all intersections of drone A and drone B
2 for  $i \in I$  do
3   Compute  $\Delta t_A$  and  $\Delta t_B$ 
4   if  $\Delta t_A + A.t - \Delta t_B + B.t \leq \epsilon$  then
5     Return True
6   end
7 end
8 Return False

```

Algorithm 7 is used to check whether two UAVs A and B can collide. It checks whether the time difference to arrive at the collision point is less than or equal ϵ and returns *True* in this case and *False* if otherwise.

Algorithm 8: Collision aware Departure1(U, T, r).

```

1  $u \leftarrow First(U, T, r).i$ 
2  $a_u \leftarrow First(U, T, r).\alpha_i$ 
3  $t_u \leftarrow First(U, T, r).T_i$ 
4 The departure time of the first UAV  $t_1^1 \leftarrow 0$ 
5 List of UAVs ordered in terms of departure time.  $U_d \leftarrow [u]$ 
6 Initial departure time  $T_d \leftarrow [t_u]$ 
7 List  $D$  of all UAVs delays which is initially consists of zero for each drone Schema Init
8  $U \leftarrow U \setminus \{u\}$ 
9  $\alpha \leftarrow U \setminus \{a\}$ 
10  $T \leftarrow T \setminus \{t\}$ 
11 while  $U \neq \emptyset$  do
12    $Found \leftarrow True$ 
13   while  $Found$  do
14      $F \leftarrow First(U, T, r).i$ 
15      $u \leftarrow F.i$ 
16     for  $v \in U_d$  do
17       if  $is\_colliding(u, v)$  then
18          $Found \leftarrow False$ 
19          $D_{u.index} \leftarrow D_{u.index} + \epsilon$ 
20          $T_{u.index} \leftarrow T_{u.index} + \epsilon$ 
21         break
22       end
23     end
24   end
25    $a_u \leftarrow F.\alpha_i$ 
26    $t_u \leftarrow F.T_i$ 
27    $U \leftarrow U \setminus \{u\}$ 
28    $\alpha \leftarrow U \setminus \{a_u\}$ 
29    $T \leftarrow T \setminus \{t_u\}$ 
30   Append  $u$  in  $U_d$ 
31   if  $r \cdot a_u + t_1 - t_u + D_{u.index} \geq 0$  then
32      $t_1^1 = t_1^1 + r \cdot a_u + t_1 - t_u + D_{u.index}$ 
33   end
34   Append  $t_1^1$  in  $T_d$ 
35    $t_1 = t_u$ 
36 end
37 Return  $U_d, T_d$ 

```



Algorithm 8 addresses the collision issue, by extending Algorithm 5. This algorithm is chosen to improve collision prevention, since the discussed results shows that it outperforms other scheduling schemes.

The collision possibility is handled from Line 16 to 23 where, Algorithm 7 is called to check whether, for a next UAV to depart, there is a possible collision. In the case where a collision is possible, the UAV gets delayed for ϵ units of time.

Note that it follows from the time complexity of Algorithm 1 and Lines 13 and 17 (whose time complexity is $\mathcal{O}(U)$ each) that the time complexity of Algorithm 8 is $\mathcal{O}(U^5)$.

4.5 Simulation and Analysis

In this section the three heuristics are compared using simulation which also reveals the accuracy effect. Finally, the persistent efficiency is formally proven. In each simulation, 100 UAVs are considered where the revisit deadline is set to 60 min (one hour). The time to be taken by each UAV to arrive at the destination is randomly chosen in the interval $[1,100]$, however for the collision handling based experiences $\epsilon = 100$ min (delaying time for a colliding UAV). A case is considered where the violation of the revisit deadline and the maximization of the visiting time have the same importance (that is $\beta = \gamma$) The simulation tool is Python. The three algorithms have been implemented, and run on an i5 computer which runs on a 64 bits and processes at 3.30 GHZ.

4.5.1 Perfect system. In this subsection, a perfect system is assumed where the accuracy of all UAVs is 100% ($\alpha = 1$). For each assignment the waiting time of the target being visited is recorded.

The three algorithms are evaluated and compared using the time elapsed when the destination is waiting to be visited. The waiting time is plotted against its corresponding i^{th} deployment ($i \in \mathbb{Z}$), where the best algorithm is the one whose corresponding curve is closest to the horizontal line passing through the revisit deadline ($y = 60$).

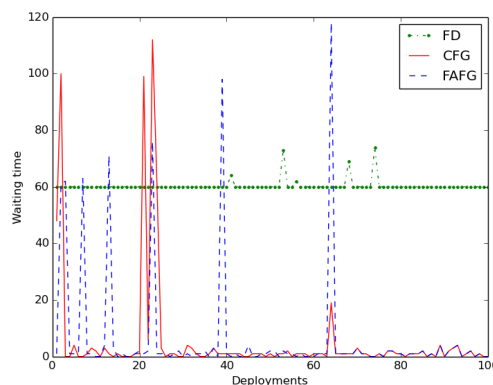


Figure 4.3: Algorithms comparison for perfect system.

Figure 4.3 compares the three algorithms. Firstly comparing CFG and FAFG, the graph shows that there is a big gap between the two plots and the line $y = 60$; both algorithms have long overdue times but the FAFG over due time is longer. For many of the deployments, the graphs are close to $y = 0$ which shows that each of the UAVs does not wait long enough to meet the revisit deadline. The similarity of the two graphs can be observed from the 66th deployment up to the last deployment.

Figure 4.3 shows that for the FD algorithm, there are only five cases where the deadline has been missed. The maximum overdue time for FD is over 15 minutes whereas the total overdue time is 40 min. The

curve is the closest to the line $y = 60$ compared to the two other algorithms and this shows a best performance of the algorithm. This is because the choice in FD takes care of the revisit deadline whereas the other algorithms do not.

4.5.2 Effect of the UAV's accuracy (α). The accuracy percentage of UAVs is assumed to be normally distributed in the 100 UAVs, with mean m and standard deviation std . That is,

$$\alpha \cdot 100 \rightsquigarrow \mathcal{N}(m, std).$$

Three cases are presented:

1. $\alpha \cdot 100 \rightsquigarrow \mathcal{N}(60, 5)$: this is a reference case where the accuracy is medium.
2. $\alpha \cdot 100 \rightsquigarrow \mathcal{N}(90, 5)$: this is the case where the average is increased while keeping the standard deviation the same. Here, the accuracy of all UAVs is likely to be increased while the average of their difference in accuracy remains unchanged.
3. $\alpha \cdot 100 \rightsquigarrow \mathcal{N}(90, 2)$: in this case, only the standard deviation changes. This corresponds to the case where, the difference in accuracy is small whereas the expected accuracy for all is constantly 90%.

The three cases are presented in Figures 4.4, 4.5 and 4.6, respectively.

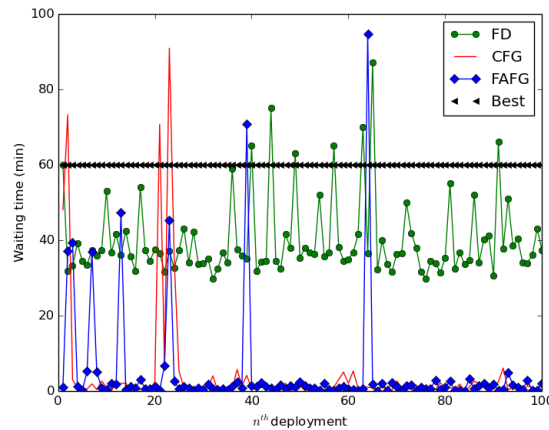


Figure 4.4: Comparison for the case where $\alpha \cdot 100 \rightsquigarrow \mathcal{N}(60, 5)$.

In this case the accuracy is expected to be 60%, with a standard deviation of 5, Figure 4.4, shows that the expected total waiting time per round for *FD* is 40 min whereas for the other algorithms, it is close to 3.

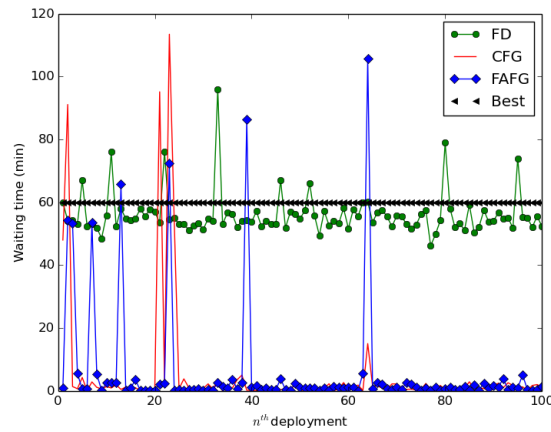


Figure 4.5: Comparison for the case where $\alpha \cdot 100 \rightsquigarrow \mathcal{N}(90, 5)$.

Figure 4.5 shows that the increase of the accuracy to 90% brings the results found in Figure 4.4 closer (from below) to the best result, keeping the same order as before (see Figure 4.4). This is why the number of deadline misses increases for all the algorithms.

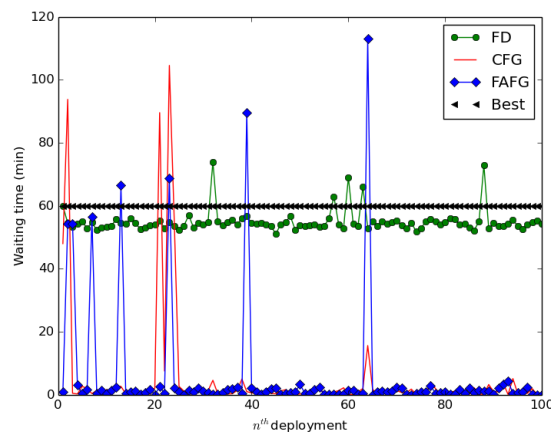


Figure 4.6: Comparison for the case where $\alpha \cdot 100 \rightsquigarrow \mathcal{N}(90, 2)$.

Figure 4.6 shows that reducing the difference in UAVs accuracy (standard deviation) increases the chance of having the waiting time less but close to 60 minutes, for all the three algorithms. For all the algorithms, the number of deadline misses decreases because the majority of the cases correspond to stale visits, and a reduction in standard deviation reduces the difference in waiting time range.

4.5.3 Analysis: persistence handling. The previous result is based on a single round (k^{th} visitation for each UAV). In this subsection, the persistent effectiveness of the three heuristics is proved.

The next visitation

The case where all UAVs need to visit a single destination using an optimal schedule has already been considered. Given one round X_i , the next round (X_{i+1}) schedule can be determined by the last visitation in X_i , and the UAVs which have completed the return flight (the available ones), are the ones ready for the next round assignment.

4.5.4 Proposition. Conservation of the assignment order. Given a set UAVs ordered in terms of ascending departure time, if the ordering is the output of one of the three algorithms, then for each round, the optimal assignment order is conserved.

Proof IV-C.

Consider the set $U = \{u_1, u_2, \dots, u_k, u_{k+1}, u_{k+2}, \dots, u_n\}$ of UAVs aligned on a timeline depending on the time t_i needed for flying to the target. Figure 4.7 shows this alignment and the letter r symbolises the required target revisit deadline.



Figure 4.7: Ordered UAVs on a timeline.

There are the following three cases.

1. **When the applied algorithm is FAFG.** In this case the UAVs depart in the same order as shown on the timeline (see Figure 4.7). Since for any two UAVs, the first to go is the one using the smallest time to arrive at the destination, the first to return back and arrive at its initial position will be the first to have left.
2. **When the applied algorithm is FD.** In this case the UAVs are ordered as in the list

$$U = \{u_k, \dots, 2, 1, u_{k+1}, u_{k+2}, \dots, u_n\}.$$

It is necessary to show that, for any two UAVs u_i and u_j in the list U , the first to depart is always the first to return to its initial position, and hence be available for the next round assignment. Let $t_i < t_j$

- If $t_j < r$, then u_j is to fly before u_i . In this case u_i will arrive at the target after spending the time $t \geq r$ after the visitation of u_j . Since $t_j < r$, it follows that $t_j < t$ and hence u_j arrives at the target after the arrival of u_j at its initial position, and thus u_j is the first to be available for the next round assignment.
- If $t_j > r$ or $t_i > r$, then u_i is the first to depart. Since $t_i < t_j$, u_i will be the one to arrive back at its initial position first and hence the first to be available for the next round assignment.

3. **When the applied algorithm is CFG.** The proposition can be proved in exactly the same way where the time r (see Figure 4.7) is replaced by the average arrival time of all UAVs. That is

$$r = \frac{1}{n} \sum_{t_i}^{t_n} t_i.$$

NOTE: Since the assignment order is conserved (see Proposition 4.5.4), and the flying time of all UAVs is constant, the system can be monitored using a central unit. The total time spent by all UAVs in one round T is considered to be the period of each UAV departure. So, in this case, the UAVs do not need to organise themselves ignoring communication (given that either collision is not possible or that otherwise the wireless communication is used for UAVs collision avoidance). The system only requires the initial

set up, where each UAV, say i has its own trajectory, and knows its first flight time t_i^d . Its flight time in the k^{th} round is inductively $t_i^d + (k - 1)T$. Here, given the departure time of the first UAV to be t_1^d , the flight time of the last UAV to be t_n^d and the time spent by the last flight to be $2t_n$ (the return time), the total time is

$$T = t_n^d - t_1^d + 2t_n.$$

4.6 Effect of the revisit deadline (r)

The distribution of the UAVs times to arrive distributed as is shown in Figure 4.8.

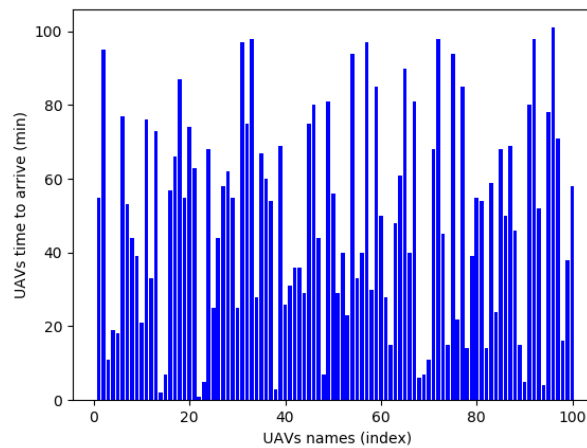
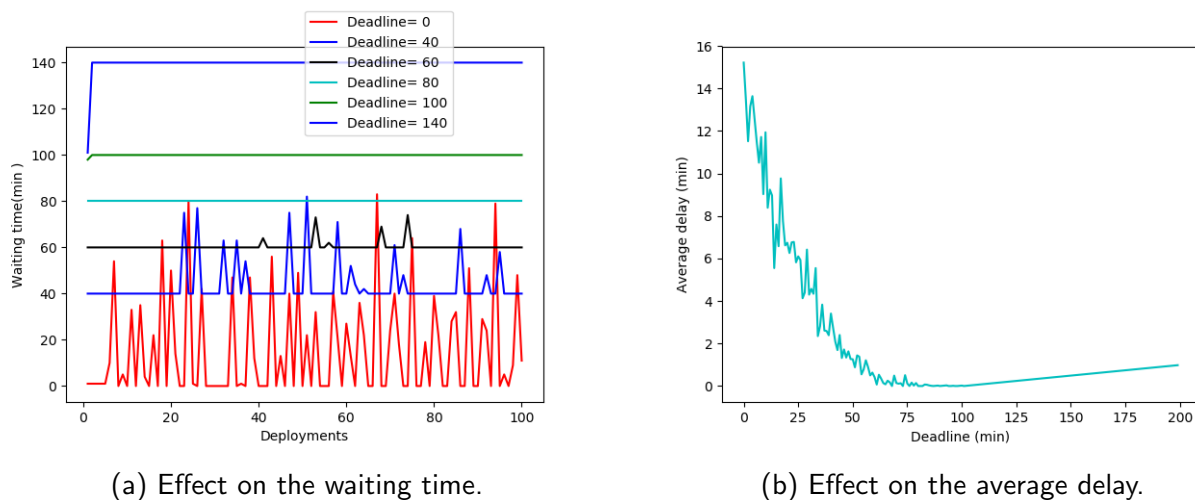


Figure 4.8: UAVs arriving time.

Here the effect on the revisit deadline on waiting time is considered and the average delay corresponding to deadlines is deduced. This has been done by changing the deadline values and recording the corresponding delay. This course of action has been motivated by the fact that the deadline which corresponds to the lowest delay could be computed.



(a) Effect on the waiting time.

(b) Effect on the average delay.

Figure 4.9: Waiting time and average delay.

Figure 4.9a shows the waiting time for every UAV deployment, for different values of the deadline. The figure shows that the lower the deadline, the more missing deadlines happen. The figure shows that

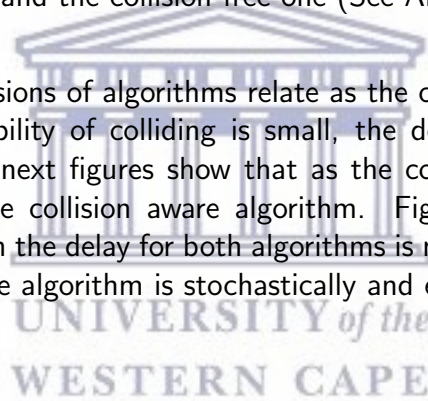
with a revisit deadline of 80 min, there is absolutely no deadline missing. This case corresponds to a case where the first UAV to visit the target takes exactly 80 min to arrive, and the remaining UAVs take different (from 80) or the same number of minutes to arrive at the target. With higher deadlines, the figure shows only one missing deadline case. This corresponds to the case where the first UAVs time to travel to the target is different from the value of the deadline and all remaining UAVs can take a shorter (than the deadline) time to travel to the target.

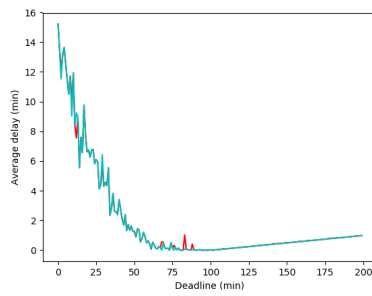
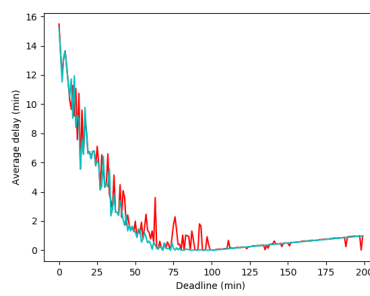
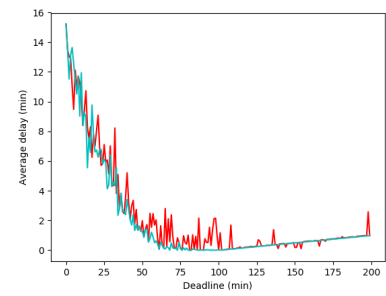
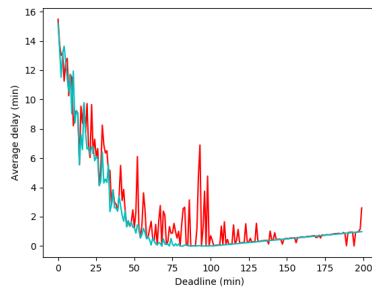
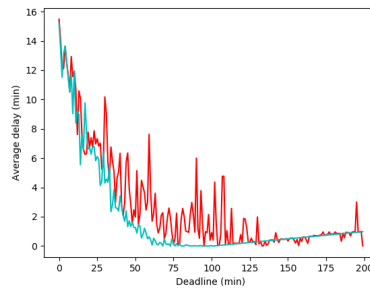
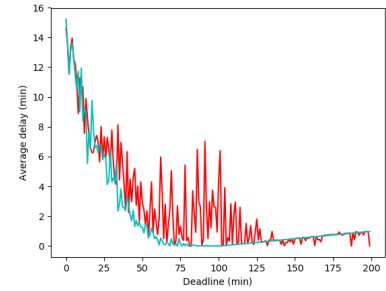
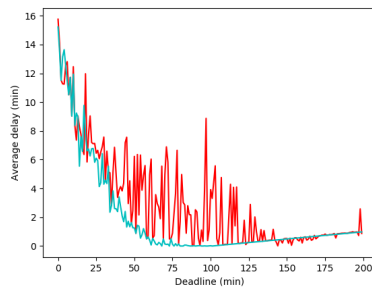
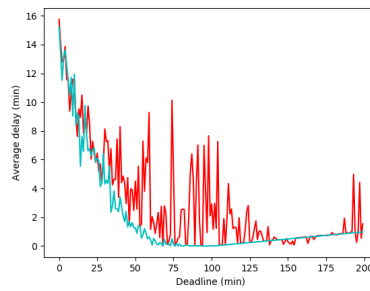
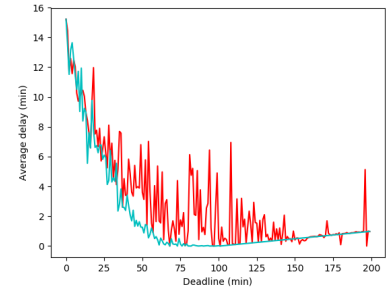
Figure 4.9b shows the average delay of all UAVs per round. Here an n^{th} round is completed when every UAV has visited the target for the n^{th} time. The figure shows that for deadlines less than 80min, the delay is stochastically and quickly decreasing. For higher deadlines, the average delay increases slowly. This is because for higher deadlines, the first UAVs will miss the deadline and the corresponding average delay will be affected accordingly.

4.7 Delay comparison

In this section a general case is assumed where UAVs may nearly or precisely collide at a finite number of points. It is assumed that these point can be computed and hence the probability of collision for each UAV can be computed and is approximately same for all UAVs. The times distribution as shown in Figure 4.8 is used here. For various probabilities, the average delay has been computed for the collision aware algorithm (See Algorithm 8) and the collision free one (See Algorithm 5) (the one assuming that the collision does not happen).

Figure 4.10 shows how the two versions of algorithms relate as the collision probability changes. Figure 4.10a shows that when the probability of colliding is small, the delay for both algorithms is almost the same for every deadline. The next figures show that as the collision probability is increased, the expected delays gets bigger for the collision aware algorithm. Figure 4.10i shows that with a high collision probability, the difference in the delay for both algorithms is not very big. On the other hand the figures show that the collision aware algorithm is stochastically and expectedly higher than the collision free algorithm.



(a) $Probability = \frac{1}{11}$.(b) $Probability = \frac{11}{21}$.(c) $Probability = \frac{21}{31}$.(d) $Probability = \frac{31}{41}$.(e) $Probability = \frac{41}{51}$.(f) $Probability = \frac{51}{61}$.(g) $Probability = \frac{61}{71}$.(h) $Probability = \frac{71}{81}$.(i) $Probability = \frac{81}{91}$.UNIVERSITY of the
WESTERN CAPE
Figure 4.10: Load balancing.

4.8 Conclusion

In this chapter, a team of UAVs is monitored to persistently visit one fixed target taking into consideration of the revisit deadline constraint. The problem has been mathematically expressed as a dynamic assignment problem and hence proved to be intractable. Heuristics to solve the problem have been proposed to minimize the cost corresponding to maximization of the total visiting time and minimizing the chance of not meeting the deadline. Using simulation, the three heuristics have been compared. The persistent effectiveness of the heuristics has been proven. The focus of this chapter has been on the management of an airborne network of UAVs for public safety deployment scenarios. When endowed with sensor or gateway devices, such a network of UAVs can be used as an airborne sensor network tasked to collect environmental data and sensor readings from a ground-based sensor network. The management of the resulting heterogeneous network of drones and ground-based sensor devices is a complex task that can benefit from the redesign of the traffic engineering techniques proposed in [110, 111] for QoS support.

5. Multiple Target Coverage

Data muling using UAVs/drones is currently emerging as an alternative to the traditional traffic engineering techniques used in wireless sensor networks, when wireless communication is not an option or the least cost-efficient solution. This chapter revisits the issue of traffic engineering in Internet-of-Things (IoT) settings, to assess the relevance of using UAVs for the persistent collection of sensor readings from the sensor nodes, which are located in a restricted environment and their delivery to base stations where further processing is performed. Under consideration here is a persistent path planning and UAV allocation model, where a team of UAVs coming from various base stations are used to collect data from ground sensors and deliver the collected information to their closest base stations. This problem is mathematically formalised and proven to be NP-hard. A heuristic solution for the problem is proposed and its relative efficiency is evaluated through simulation.

5.1 Introduction

Unnamed Aerial Vehicles (UAVs) are emerging as a flexible and cheap alternative to traffic engineering techniques which have been traditionally used in IoT settings, to transport sensor readings from their points of collection to their processing places. However, the joint path finding and resource allocation for a team when tasked to achieve collaborative data muling is still an issue that require further investigations. Furthermore, while accurate solutions to data muling problems are still scarce, especially when considering the limited flying autonomy of the battery-powered UAVs, issues related to the efficient task allocation to a team of UAVs under stringent data collection requirements still need to be addressed.

Potential applications of the proposed model include (i) city surveillance in order to evaluate risks and respond with appropriate actions by having a team of UAVs persistently/permanently visiting locations of interest in a smart city for public safety, (ii) parking spots localization [112] (iii) pollution monitoring [113] ; (iv) drought mitigation to support small scale farming in rural areas[114, 115] by using a team of UAVs to collect farmland image collection and processing these images to achieve situation recognition for precision irrigation; (v) periodic surveillance of buildings and cities' infrastructures for structural health monitoring and maintenance; and (vi) the extension of the reach of community mesh networks in rural settings for healthcare [116, 117] by using a team of UAVs (such as drones) as wireless access points.

Sensors visitation under the fuel consumption constraints has been addressed in [99], and the visitation under the revisit deadline constraint has been proposed in [100]. Both studies assume a single moving agent (UAV) which optimally visits various targets. [34] proposes a cooperative UAVs model where many targets are visited by a team of UAVs for persistent surveillance and pursuit. In this study, the UAVs do not communicate with each other but rather rely on the information from the static underground sensors, which are optimally placed as proposed in [35]. However, none of these models consider the persistent data delivery and heterogeneity of UAVs which might have different fabrics and characteristics. Furthermore, neither the energy/battery consumption while the UAVs are waiting for the updated information from the terrestrial sensor network, nor the penalty associated with stale information due to late visitation by the UAV to the sensor nodes, have been accounted for. While models have been proposed in [118, 119, 120, 121] for the periodic and persistent UAVs visitation of a single target from different positions, the models do not consider the path planning issues, and this is required necessary for restricted environments.

This chapter proposes a persistent path planning and task allocation model where, a team of UAVs coming from various base stations are used to collect sensor readings from ground sensors and deliver the collected information to their closest base stations. The underlying data muling problem is i) mathematically

formalised as a constrained optimisation problem, ii) proven to be intractable and iii) solved using a heuristic solution, whose relative efficiency is proven through simulation modelling.

The rest of this chapter is organised as follows. The cooperative data muling model is presented in Section 5.2.2 and its algorithmic solution provided in the same section. Simulated results are provided and discussed in Section 5.3 while the conclusion is drawn in Section 5.4.

5.2 The Cooperative Data Muling Model

In this chapter, an Internet-of-Things in Motion model depicted by Figure 5.1 is considered. It is assumed that UAVs are assisted by special ground-based sensors which locally collect data from other sensors. That is, sensors are grouped into separated clusters, each with its own sink node (the cluster head), where the information is to be collected from other sensor nodes (cluster members) and relayed to UAVs which deliver the sensor readings to base stations. Note that only cluster heads can communicate with UAVs, and the optimal clustering scheme is not covered in this chapter. Furthermore, the inner cluster communication technology is not covered here (it has been discussed in [81]).

5.2.1 System view. The system is shown in Figure 5.1.

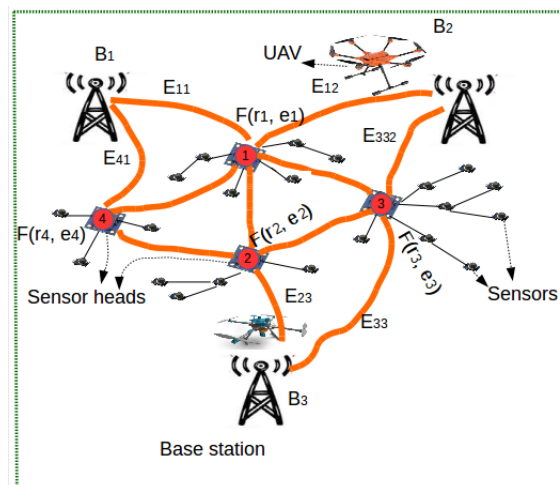


Figure 5.1: Cooperative Data Muling.

The cooperative data muling model considered in this chapter is illustrated by Figure 5.1, which reveals three base stations (B_1 , B_2 and B_3) from which UAVs take off to collect data from sinks located in a region of interest. In this illustrative scenario, all possible collection paths which can be taken by each UAV from the base stations to access data collected by sink nodes (1, 2, 3 and 4) and deliver the collected data to the closest base stations, are represented by thick (orange) lines. Here, it is assumed that UAVs are assisted by special nodes (cluster heads/sinks) to collect information to reduce the loss due to UAVs constraints (fuelling, timing, etc.). Furthermore, it is assumed that the UAVs paths topology is known (this means that all possible paths between nodes are known). The total energy required for data collection at a node, say i , is a function $F(r_i, e_i)$ of its revisit deadline r_i (the time for which the node has to be revisited) and the energy e_i required to transfer data from node i to a UAV. The travel cost from a base station B_j to a sink i is translated in an energy metric denoted by E_{ij} , and the transportation cost between two sinks i and k is also translated into an energy metric denoted by e_{ik} .

5.2.2 The Data Muling Problem. In this subsection, the problem is modelled as a constrained optimisation problem. Firstly all cost related terms are defined and later, these are combined to form a cost function.

UAVs waiting time on sink nodes

Let t_i be actual time spent by an UAV to arrive at the sink i from a base station and r_i be the expected time for the UAV to arrive at the sink i . This is also referred to as *revisit time* at collection point i . The sink visit assumptions may be expressed in terms of penalties/costs for early/late visits to the sink nodes, the collection of information and a risk associated with the autonomy of the UAVs. These costs are described as follows

- **Early visit penalty:** An early visit penalty will be assigned to an UAV if $t_i < r_i$ to express the case where the visiting UAV arrives earlier than expected. In this case, the UAV will wait for a period of time $w_i = r_i - t_i$ needed by the sink node to capture information from the field and transmit it to the waiting UAV.
- **Late visit penalty:** A late visit penalty will be applied to the UAV if $t_i > r_i$ to express the fact that the visiting UAV is late by a period of time $l_i = t_i - r_i$ that has been wasted by the UAV to arriving late at a collection point where data was ready for collection. This penalty can also be expressed using a piece-wise function.
- **Data collection cost:** A data collection cost will be applied to any UAV to account for the fact that the UAV has to use the energy e_i to collect information from the visited node i . Note that while the costs w_i and l_i depend on how the terrestrial and airborne sink networks have been traffic engineered, the data collection cost e_i may depend on different engineering parameters and functions which may be bound to the communication interfaces of the equipment used by both the ground sink nodes and the UAVs and the protocol used for such communication.

For each UAV, the total cost $F(r, e_i)$ of visiting the sink i , without taking into account the travelling cost is expressed by

$$F(r, e_i) = \alpha w_i u(w_i) + \beta l_i u(l_i) + \gamma e_i, \quad (5.2.1)$$

where $u(w_i)$ and $u(l_i)$ are the values of a unit step function applied to w_i and l_i , respectively. The coefficients α , β , and γ are associated with the importance/weighting allocated to the early and late arrival penalties and the data transfer penalty respectively.

The assumed network

Under consideration here is a hybrid sensor network represented by a bidirected graph $\mathcal{G} = (\mathcal{S}, \mathcal{N}, \mathcal{L}, \mathcal{B}, \mathcal{P})$, where \mathcal{S} is the set of all sensor nodes, $\mathcal{N} \subset \mathcal{S}$ is the set of all sinks, \mathcal{L} is the set of wireless communication links between the sensor nodes, \mathcal{B} is the set of UAV base stations while \mathcal{P} is the set links showing possible moves of UAVs. Here, a move expresses one of two kinds of connection:

- **Base station-sink :** These are bidirectional paths connecting sink and base stations. The cost of moving from a base station b to a sink i is denoted by E_{bi} and its opposite is E_{ib} , with $E_{ib} = E_{bi}$
- **sink-sink :** These are the paths connecting sinks and the cost to move from one sink i to j is denoted by e_{ij} , with $e_{ij} = e_{ji}$

Initial condition

- Each UAV is assumed to start its journey from a base station, where it can make the first visit without colliding with another UAV at the same sink node.

- The waiting times at the base stations are assumed to be zero. That is $l_i = w_j = 0, \forall i, j \in S$

Here, it is assumed that the maximum number of UAVs at each base station is equal to the degree/capacity of the base station.

The data muling modelling

The data muling is performed in two steps

- **Data collection.** During data collection, an UAV is to move from a Base station a to collect data from k sinks labelled by a set of indices $p^* = [1, 2, \dots, k]$. In this case, the path used for data collection is represented by $p = [a, 1, 2, \dots, k]$. The energy required for this step is expressed by

$$C(p) = E_{a1} + \sum_{i \in p^*} F(r, e_i) + \sum_{i, j \in p^* \wedge j = i+1} e_{ij}. \quad (5.2.2)$$

- **Data delivery.** During data delivery, an UAV may pass by some already visited sink to deliver information to its closest base station b . In this case, the corresponding energy is expressed as $E(p)$.

Therefore, the total energy required for data collection and delivery is given by the equation

$$C(p) + E(p) = E_{a1} + \sum_{i \in p^*} F(r, e_i) + \sum_{i, j \in p^* \wedge j = i+1} (e_{ij}) + E(p). \quad (5.2.3)$$

The data muling problem consists of finding an optimal path for each UAV so that the total energy expended by all the UAVs to collect and deliver the sensor readings/data without colliding is minimized. Mathematically, the set of UAVs is represented by $U = \{1, 2, 3, \dots, m\}$, where each UAV say u departing from base station a_u will follow path p_u to collect data at locations of interest and another path (maybe different from p_u) to deliver the data to its closet base station. Considering 1_u , the first sink to be visited by the UAV u . The data muling problem is formulated as follows.

$$\min Z = \sum_{u=1}^m \left(E_{a_u 1_u} + \sum_{i \in p_u^*} F(r, e_i) + \sum_{\substack{i, j \in p_u^* \\ j=i+1}} (e_{ij}) + E(p_u) \right), \quad (5.2.4)$$

Subject to,

$$\forall v, w \in U, p_v^* \cap p_w^* = \emptyset = d(p_v^*) \cap d(p_w^*) \quad (5.2.4a)$$

$$\bigcup_{u \in U} p_u^* = S \quad (5.2.4b)$$

$$e_i, e_{ij}, E(p), r, E_{a_u 1_u} \geq 0, \forall i, j, p, a_u, 1_u. \quad (5.2.4c)$$

The first constraint (Equation 5.2.4a) states that any two collection or delivery paths have no sink in common. This guarantees collision avoidance for the UAVs. On the other hand, the second constraint (Equation 5.2.4b) expresses the fact that all sinks are to be visited. The last constraint (Equation 5.2.4c) shows that all variables are positively valued.

5.2.3 Constrained visitation. Consider Equation 5.2.1. In given scenarios, the variables have got very strict conditions and instead of being part of the cost function, they need to be part of the problem constraint. Table 5.1 shows all possible models. Here a "1" shows the case where a corresponding variable is restricted (part of the constraints) and a "0" shows where it is not.

Table 5.1: Data collection scenarios.

Waiting penalty (w_i)	Late penalty (l_i)	Explanation
0	0	None of the two variables is bounded
0	1	Only the late penalty is bounded
1	0	Only the waiting penalty is bounded
1	1	Both variables are bounded

The optimisation problem becomes changed by its constraints so as to become as follows.

$$\min Z = \sum_{u=1}^m \left(E_{a_u 1_u} + \sum_{i \in p_u^*} F(r, e_i) + \sum_{\substack{i, j \in p_u^* \\ j=i+1}} (e_{ij}) + E(p_u) \right), \quad (5.2.5)$$

Subject to,

$$\forall v, w \in U, p_v^* \cap p_w^* = \emptyset = d(p_v^*) \cap d(p_w^*) \quad (5.2.5a)$$

$$\bigcup_{u \in U} p_u^* = S \quad (5.2.5b)$$

$$e_i, e_{ij}, E(p), r, E_{a_u 1_u} \geq 0, \forall i, j, p, a_u, 1_u. \quad (5.2.5c)$$

$$0 \leq w_i \leq W_i \quad (5.2.5d)$$

$$0 \leq l_i \leq L_i \quad (5.2.5e)$$

where W_i and L_i are the predetermined thresholds which may take any non negative value.

5.2.4 Related problems and solutions. The data muling problem considered in this chapter is closely related to the file recovery problem in [122] (NP-hard problem) solved by curving techniques, including those using the Parallel Unique Path (PUP) algorithm. This problem considers a case of many fragmented files which need to be reassembled, starting from their headers, which are assumed to be known initially. The PUP algorithm is a variation of Dijkstra's routing algorithm[52] where starting from the headers, clusters are successively added based on their best matches.

This is done with the aim of building paths from headers having a cluster added to an existing path if and only if the link to it has the least weight. On the other hand the Vehicle Routing Problem (VRP)[49, 54] consists of finding the optimal road from a depot, to be taken for delivering resources to customers and coming back to the depot. Exact and heuristic algorithms for its solution have been surveyed in[49]. In the survey, all stated algorithms assume a single distance matrix (the cost matrix) and hence could fail to be a good fit for the current persistent visitation scenario since in this case the weighting of nodes matters and does not take a fixed value. Furthermore, for the VRP, vehicles end their trips at the depots where they started from. This would limit the number of topologies where the data muling problem is solvable and also could impose a data muling scheme which is not necessarily optimal. Note that in the current case, it is important that late and stale visitations are taken care of, and this depends on the dynamic position of UAVs (see the Equation 5.2.1). Furthermore, UAVs deliver the collected information to optimal base stations (which are not necessarily their starting points).

5.2.5 The data muling problem intractability. To prove its intractability, a polynomial reduction of one-depot VRP is provided (which is known to be an NP-hard problem), into a special case of the data muling problem: the case where each sinks weight is zero. The transformation consists of a two-step process which transforms the graph \mathcal{G} as follows.

- a. Group all Base stations in \mathcal{S} in one cluster/group and consider this cluster/group as a special node for the graph, this gives the VRP's topology $\mathcal{G}' = (\mathcal{S}, \mathcal{N}, \mathcal{L}, \mathcal{B}', \mathcal{P})$, where $\#\mathcal{B}' = 1$.
- b. For every link of \mathcal{G}' , make the link weight in the new graph (found in a.) the inverse of the weight in the Graph \mathcal{G}' .

This will reduce the VRP's into the data muling problem's solution.

Clearly, the time complexity of the transformation process is polynomial since Step *a* has a complexity $\mathcal{O}(\#\mathcal{S})$ and Step *b* has complexity $\mathcal{O}(\#\mathcal{R})$. The time complexity for the whole graph transformation/reduction process is therefore $\mathcal{O}(\#\mathcal{R}) + \mathcal{O}(\#\mathcal{S})$, which is polynomial. This shows that the problem of interest in this chapter is NP-hard and hence a heuristic solution is important.

5.2.6 The Data Muling Algorithm. In this chapter, Dijkstra's algorithm is adapted, in the same way as it is done in [122], to solve the data muling problem. While many rounds are considered by this authors algorithm, only the case where each node is visited only once per round is considered. It is assumed that each UAV is capable of collecting and delivering data to a base station where it can be recharged, before going for another data collection round.

Algorithm 5.2.10 has two major steps: the first step consists of using Equation 5.2.2 to select the best node to visit for every UAV (Step 6-7); the second step consists of adjusting the UAV's paths by choosing the cheapest UAV for every best node (Step 8-23). Once the visitation is done, the collection paths are captured in $Cpaths$ and the two steps are repeated to select the nearest base station for every UAV data delivery following the delivery paths recorded by $Dpaths$.

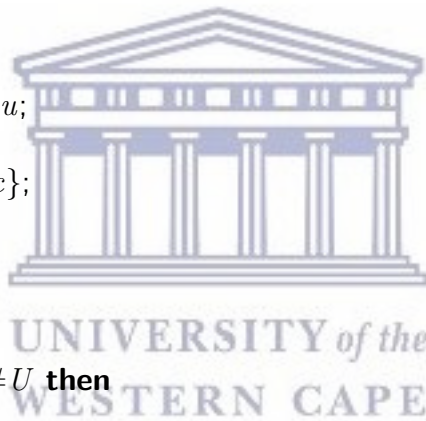
5.2.7 Proposition (Polynomial termination). Algorithm 5.2.10 terminates in polynomial time, when all sink nodes have been visited.

Algorithm 9: Cooperative algorithm.

```

1 Assume a network  $G(N,L,B,P)$  as specified in Section 5.2.2;
2  $Choices \leftarrow$  all sink to be visited ;
3 Initialise the path to the initial hosting base stations;
4  $done \leftarrow choices \cup B$ ;
5 while  $choices \neq \emptyset$  do
6   for  $u \in U$  do
7     | Select the next destination of least cost, using Equation 5.2.2 ;
8   end
9    $Assign = \emptyset$ ;
10  for  $u \in U$  do
11    | Let  $c$  be the choice of  $u$ ;
12    for  $v \in U \setminus Assign$  do
13      | if  $u$  and  $v$  selected the same choice  $c$  then
14        | Include  $c$  in the path of a UAV of least cost (5.2.2);
15        | Include the UAV in  $Assign$ ;
16        | Include the  $c$  in  $done$ ;
17        |  $Choices \leftarrow Choices \setminus \{c\}$ ;
18        | Break;
19      | end
20    | end
21    if  $c \in Choices$  then
22      | Include  $c$  in the path of  $u$ ;
23      | Include the  $c$  in  $done$ ;
24      |  $Choices \leftarrow Choices \setminus \{c\}$ ;
25      | Include  $u$  in  $Assign$ ;
26      | Break;
27    | end
28  | end
29  if  $Choices = \emptyset$  or  $\#done = \#U$  then
30    |  $Choices \leftarrow B$ ;
31    |  $done \leftarrow \emptyset$ ;
32    |  $B \leftarrow \emptyset$ ;
33    | if  $\#done \neq \#U$  then
34      |  $Cpaths \leftarrow$  all UAVs' paths;
35    | end
36  end
37  $Dpaths \leftarrow$  all current UAVs' paths ;
38 Return  $Cpaths$  and  $Dpaths$ 

```



5.2.8 proof. Note that each time a sink is included in a path of one of the UAVs, it gets excluded from the list *choice* (Lines 16 and 21). Since all UAVs paths consist of a connected graph, whenever $choice \neq \emptyset$, there is at least one UAV which makes a new selection of a next sink to visit on Line 6. So all sinks are visited.

Once $choice = \emptyset$, the next destinations become the base stations and the set $done = \emptyset$ (Lines 25 and 26 consecutively). The next step is to make $\#done = \#U$ true by assigning to every UAV a base station. In this case the statement at Line 24 is true and the set $choice = B$ which had been updated

to \emptyset . This makes Algorithm 5.2.10 stop.

On the other hand, the time complexity of the algorithm is clearly $\mathcal{O}((\#U)^2)$ which is a polynomial.

Hence the result follows.

5.2.9 Remark (Persistent visitation model). Since each UAV's computed path is ended at a base station and the UAV paths form a static network, Algorithm 5.2.10 can be used repeatedly to make a persistent visitation.

5.2.10 Algorithm. In this chapter, Dijkstra's algorithm [52] is adopted, in the same ways as it is done in [122], to solve the problem described in Subsection 5.2.4. The algorithm runs in rounds. The case under consideration is where each node is visited only once per round.

1. For each UAV, select the best (cheapest) sensors to visit. Here, the cost to visit one node is the travelling cost plus the weight of the node.
2. For all selections, choose the cheapest one, make it the current assignment and remove the chosen sensor from the selectable ones.
3. Repeat Step 2 until no more choice is available (all sensor nodes have been selected).
4. For each UAV position assign the best base station. That is the base station to which, the travel cost/distance is least.

5.2.11 Illustration of the algorithm. The algorithm in Subsection 5.2.10 is illustrated using an example. One round of the algorithm is run step by step. In this example, consider the case were the sensor node's weight is constant. That is, $\alpha = \beta = 0$ (see the constants in Equation 5.2.1).

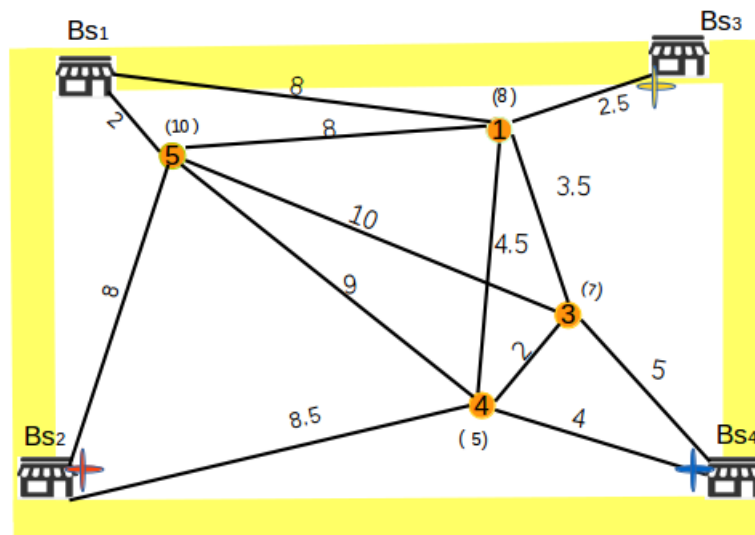


Figure 5.2: Initial step.

Figure 5.2 shows the initial conditions of the system. The system has four sensors, four base stations and three UAVs positioned at all of them except Base station Bs_1 . The links and sensors are weighted. Let u_2 , u_3 , and u_4 be the names of UAVs staying at Base Stations Bs_2 , Bs_3 , and Bs_4 , respectively.

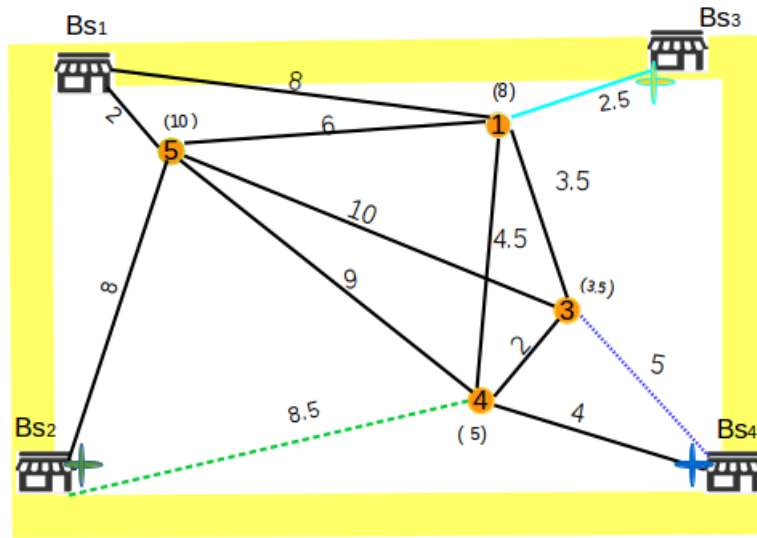


Figure 5.3: The best neighbour choice.

Figure 5.3 reveals how the UAVs make choices for their first moves. The best choice is the one corresponding to the cheapest move, which is evaluated using the weight of the road to be used together with the delay at the sensor to be visited. This is why UAVs u_2 , u_3 and u_4 move to Sensors 4, 1 and 3, respectively. At this stage, only node 5 is the only node yet to be visited.

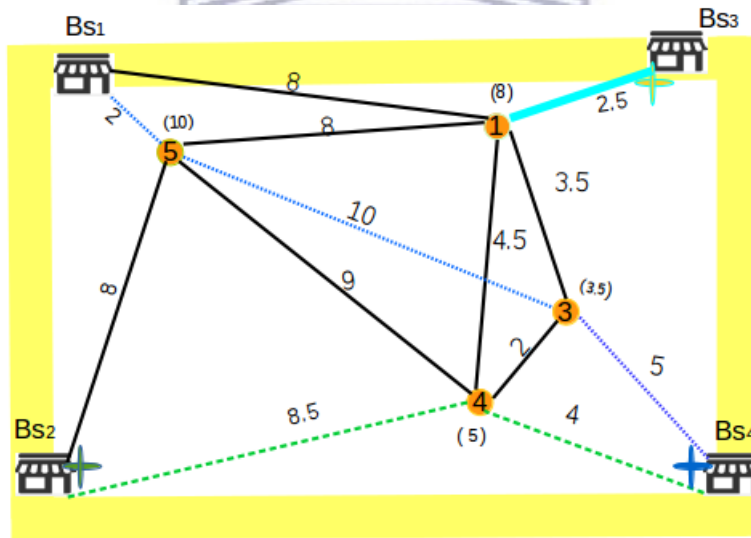


Figure 5.4: Delivery.

Figure 5.4 shows the next moves up to the end of the algorithm. It shows that UAV u_4 moves to Sensor 5, because it is the one corresponding to the cheapest move. On the other hand, the other UAVs do not have any other choice of sensor to visit. They then need to visit their closest base station. Once UAV u_4 arrives at Node 5, it visits the closest base station which is Bs_1 .

5.3 Experimental results

Python was used to run Algorithm 5.2.10 on two more complex networks (Figure 5.5). The performance of the algorithm is studied and the behaviours of considered parameters are investigated. The first

network (5.5a) consists of five base stations: B_1, B_2, B_3, B_4 and B_5 , as shown by bigger nodes in Figure 5.5a. In the figure, smaller nodes represent the sinks to be visited and the links in the network show the possible paths, the UAVs may take to visit the targets. Note here that the sink's network (network without base stations) is a complete graph (each UAV is able to move from one sink to any other one in the network, but not to any base station) where nodes are randomly deployed on a $1km^2$ area.

On the other hand, a real network is considered (Figure 5.5b) consisting of the Cape Town complete network whose nodes are police stations and their Cartesian coordinates have been extracted from GPS positions. The names corresponding to each node label are described in Chapter 2 (see Figure 2.9). Note here that in these experiments the considered UAVs are drones.

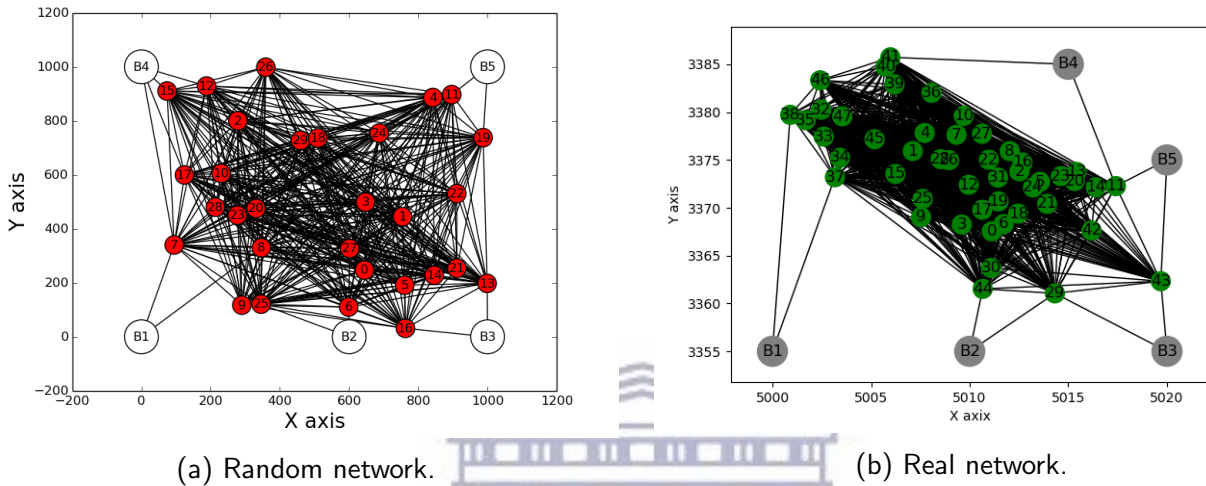


Figure 5.5: Considered networks.

As shown in Figure 5.5a, the considered network consists of nodes randomly placed in an area of size $1km^2$ and sink are labelled in terms of the energy required to collect information from them. The coordinates of nodes in both networks are in metres and could be seen or approximated using Figure 5.5. Positions (of sinks or base stations) are expressed in terms of X and Y coordinates and hence, they are presented in the 2D Cartesian coordinate system.

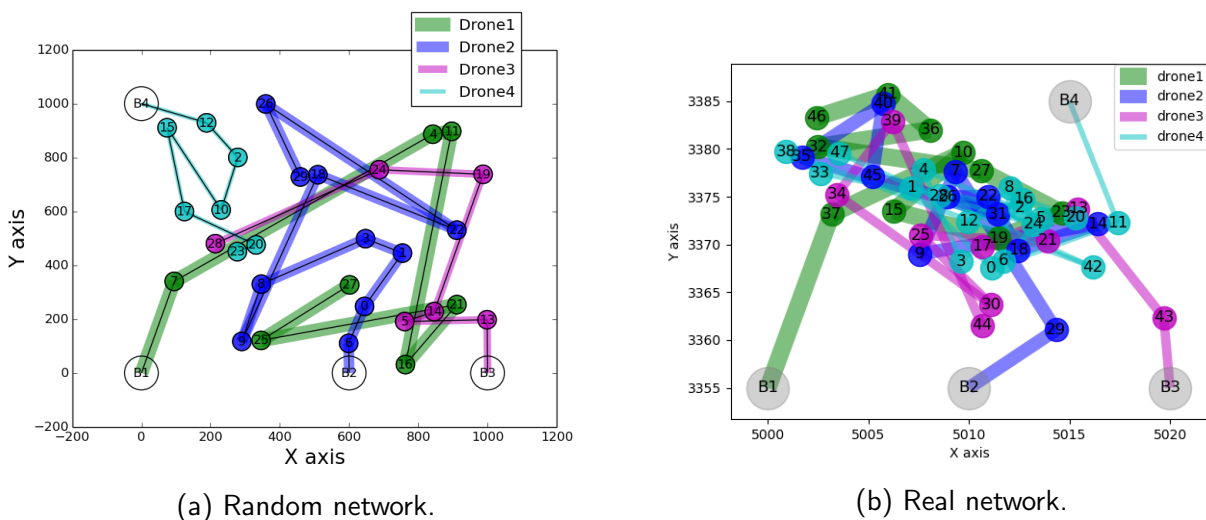


Figure 5.6: Paths generation when the UAVs have the same speeds.

5.3.1 Impact of speed distribution on path planning. The paths taken by UAVs using Algorithm 5.2.10 are presented in two steps.

Step1. Data collection: it consists of visiting all sinks using the first three steps of Algorithm 5.2.10. The corresponding path for each UAV is shown in Figures 5.6a, 5.7a and 5.8.

Step2. Data delivery: it consists of visiting a base station using the last step of the same algorithm. The results are shown in Tables, 5.2, 5.3 and 5.4, where the speed distribution of drones is also presented.

The cost function parameters are set to $\alpha = \beta = 0.5$ and $\gamma = 1$.

Figures 5.6a and 5.23e reveal that UAVs do not visit the same number of sinks. For example in Figure 5.6a, Drone1, Drone2, Drone3 and Drone4 visit 7, 10, 6 and 7 sinks, respectively.

UAV name	Speed(m/min)	Source	Returning path
Drone1	500	B1	[27, 6, B2]
Drone2	500	B2	[29, 12, B4]
Drone3	500	B3	[28, 7, B1]
Drone4	500	B4	[23, 8, B1]

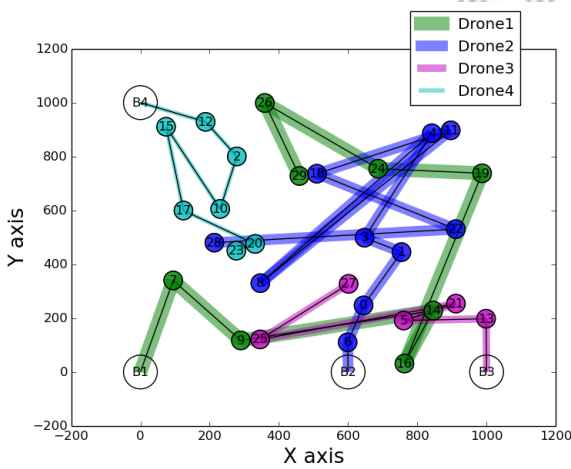
(a) Random network.

UAV name	Speed(m/min)	Source	Returning path
Drone1	500	B1	[46,41, B4]
Drone2	500	B2	[45, 11, B5]
Drone3	500	B3	[44, B2]
Drone4	500	B4	[47, 12, B4]

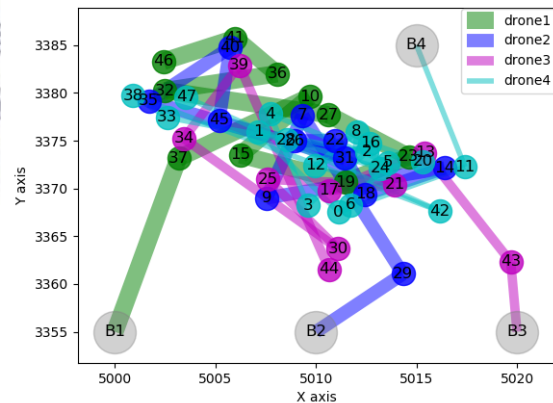
(b) Real network.

Table 5.2: Data delivery when all drones have the same speed.

Table 5.2 shows that when the speeds are the same, the delivery requires some UAVs to pass by some of the visited nodes to arrive at the base stations. This is because all sinks do not need to be connected at a base station. The table also shows that in the case of a random network, Drone3 and Drone4 deliver the collected data at the same base station (B_1), and for the real network Drone1 and Drone4 deliver the data to B4.



(a) Paths generation when UAVs have different speeds.



(b) Real network.

Figure 5.7: Paths generation when the UAVs have the different speeds.

Figure 5.7a shows that when the speeds are different, some UAVs may change their paths but others may not. For example Drone4 did not change its path between Figures 5.6a and 5.7a, whereas all the other drones did. On the other hand Figure 5.7b shows that for the real network, all drones kept their paths the same.

UAV name	Speed(m/min)	Source	Returning path
Drone1	800	B1	[29, 12, B4]
Drone2	700	B2	[28, 7, B1]
Drone3	600	B3	[27, 6, B2]
Drone4	500	B4	[23, 8, B1]

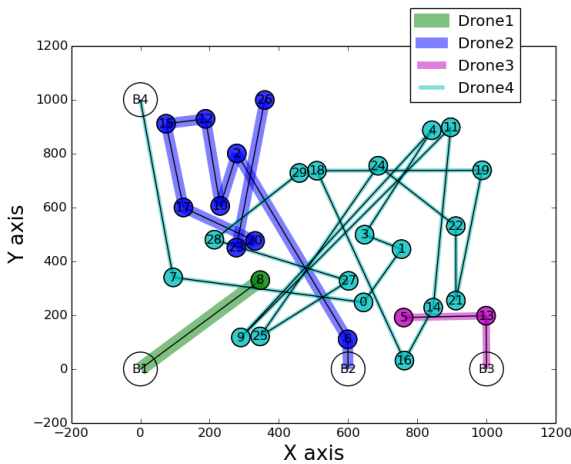
(a) Random network.

UAV name	Speed(m/min)	Source	Returning path
Drone1	800	B1	[46,41, B4]
Drone2	700	B2	[45, 11, B5]
Drone3	600	B3	[44, B2]
Drone4	500	B4	[47, 12, B4]

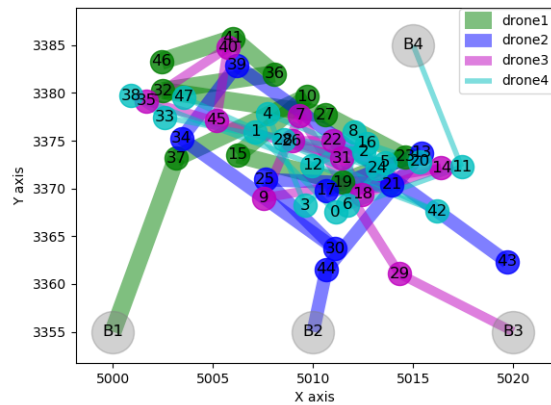
(b) Real network.

Table 5.3: Data delivery when all drones have different speeds.

Table 5.3 shows that when the drone speeds are different, the delivery also requires some UAVs to pass by some of the already visited nodes, in order to arrive at a closest base stations. Table 5.3a shows that Drone2 and Drone4 deliver the collected data at the same base station (B_1), and Table 5.3b shows that Drone1 and Drone4 deliver the collected data at the same base station (B_4).



(a) Random network.



(b) Real network.

Figure 5.8: Paths generation when speeds distribution is changed.

It is shown in Figure 5.8, that the distribution of the UAVs speeds, have an impact on paths generation. For example Figure 5.8a shows that Drone1, Drone2, Drone3 and Drone4 visit 1, 9, 2 and 18 sinks, respectively. This shows a big difference due to the fact that the choice of target base stations depends on the current and not the previous visitation costs, together with the change of UAVs speed distribution.

UAV name	speed (m/min)	Source	Returning path
Drone1	500	B1	[8, 6, B2]
Drone2	600	B2	[26, 12, B4]
Drone3	700	B3	[5, 16, B3]
Drone4	800	B4	[29, 15, B4]

(a) Random network.

UAV name	Speed(m/min)	Source	Returning path
Drone1	500	B1	[46,41, B4]
Drone2	600	B2	[43, B3]
Drone3	700	B3	[45, 11, B5]
Drone4	800	B4	[47, 12, B4]

(b) Real network.

Table 5.4: Data delivery when speed distribution changes.

Table 5.4 shows that when the speeds are differently distributed, the paths are changed and thus the delivery paths also change. Table 5.4a shows that for the random network, Drone2 and Drone4 deliver the collected data at the same base station (B_4); and for the real network, Table 5.4b shows that Drone1 and Drone4 deliver the collected data at the same base station (B_4)

For the real network, it is clear that paths did not change. This is caused by the fact that nodes are separated by a long distance and this makes very expensive for UAVs to have alternative options. Since the path generation for both networks essentially behave in the same way, only the random network is considered for the next analysis.

5.3.2 Impact of the speed distribution on the cost (total energy). The impact of the speed on the overall cost, over many runs of the algorithm is now considered. 20 runs of the Algorithm 5.2.10 are performed in three cases of speed distribution as shown in the second columns, on Tables 5.2, 5.3 and 5.4.

Same speed case

Figure 5.9 shows a case where all UAVs have a speed of 500m/min.

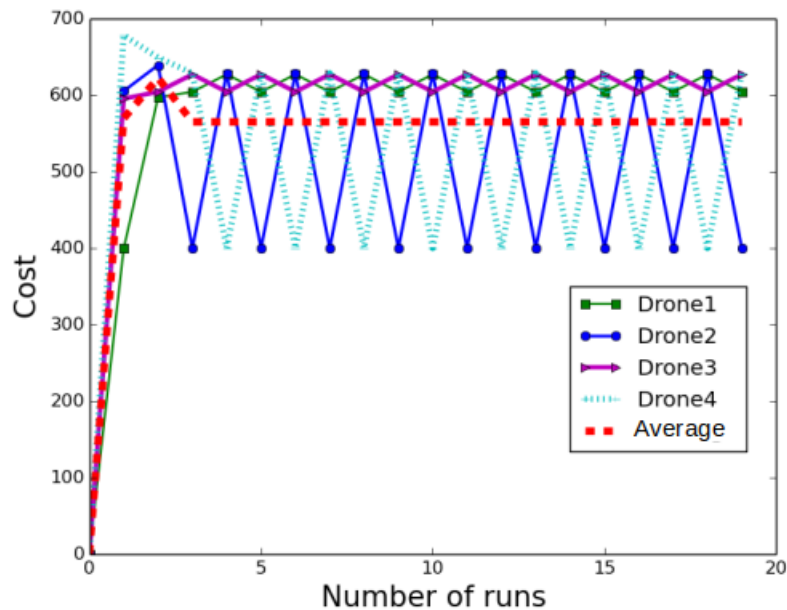


Figure 5.9: Cost at same speed.

Figure 5.9 reveals that the cost for each drone takes three or four of several fixed values. Drone4 takes four values and all the others take three and alternate with each other to take the minimum and maximum values. The average cost is constant after 3 runs and close to 560.

Different speed case

Figure 5.10 corresponds to the case where UAVs have different speed as shown in Table 5.3.

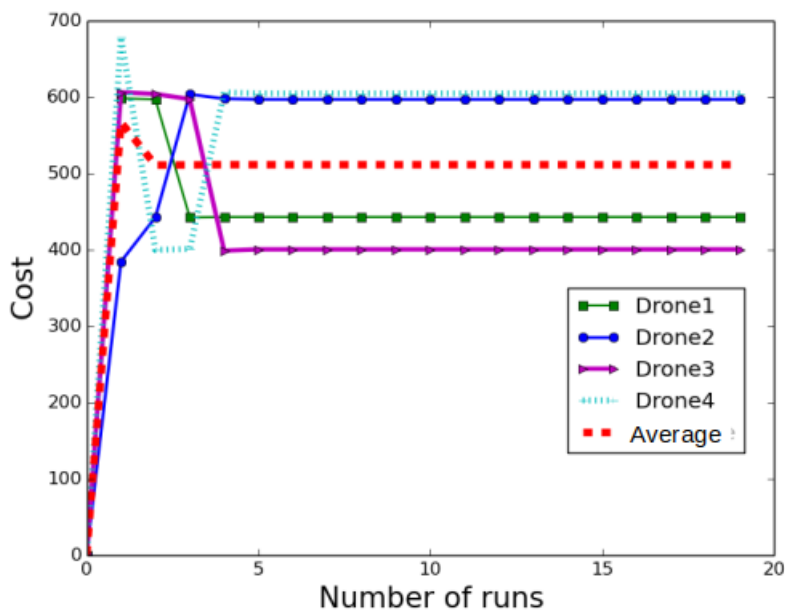


Figure 5.10: Cost at different UAVs' speeds.

Figure 5.10 shows that after the first four runs, all UAVs have constant costs where the average cost is close to 500.



Effect of speed distribution change

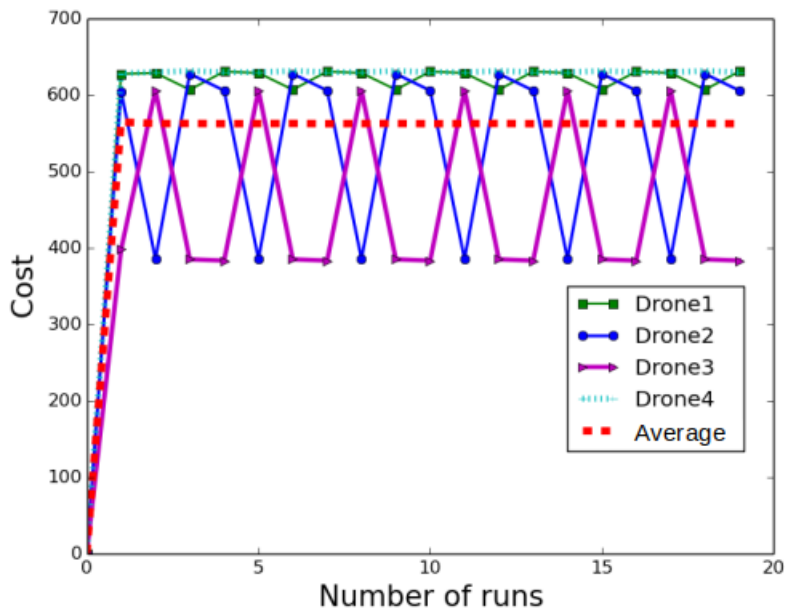


Figure 5.11: Cost related to a different speed distribution.

Figure 5.11 shows that the change in speed distribution may change the average cost and also the trends of the cost function. In this case, the amended speed distribution corresponds to the one in Table 5.4,

and shows that for each UAV, the cost changes periodically and can only take one of a few fixed values. This is why the average cost also takes one of the fixed values on a periodic basis.

5.3.3 Effect of parameters variation. In this subsection, the effect of four parameters on the cost variation as the number of runs varies is studied. The four considered parameters are described as follows.

- **Speed value:** while keeping the speed the same for all UAVs, the impact of the speed of all drones on the coverage cost is studied.
- **Overdue time (α):** the impact of overdue time on the cost is looked at. Great attention is placed on this time by incrementing the corresponding coefficient (penalty) by 0.05, for each new run.
- **Delay (β):** While all the other parameters are constant, The parameter corresponding to the delay penalty is varied in order to study how it changes the value of the coverage cost.
- **Data collection rate (γ):** Data collection rate is incremented by 0.05 for its value $\gamma = 1$, over 20 runs of the algorithm.

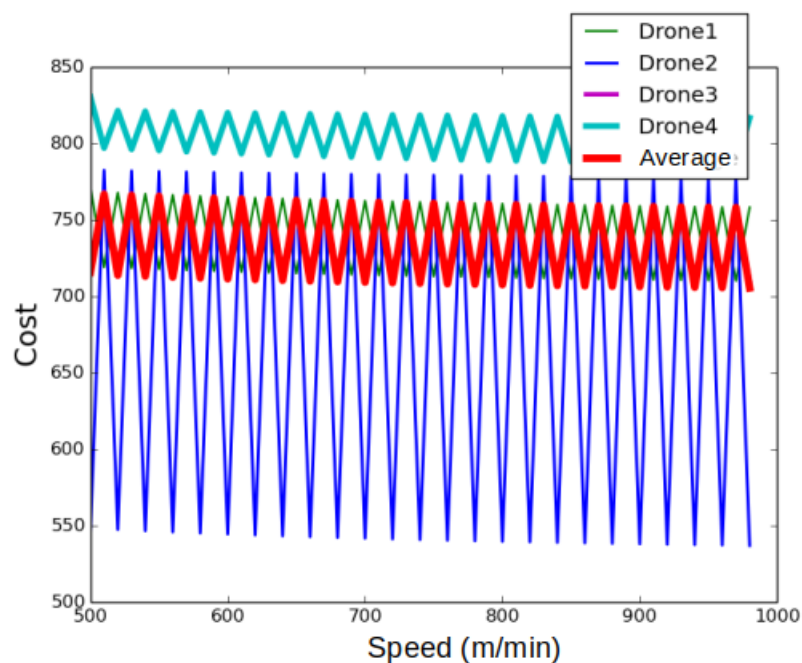


Figure 5.12: Variation of the cost with respect to the speed.

Figure 5.12 shows a case where the speed has been incremented 20 times for all UAVs. It shows that from the first run, each UAV periodically changes its cost between two cost values. Drone4 corresponds to the highest cost while notably Drone3 corresponds exactly to the average cost.

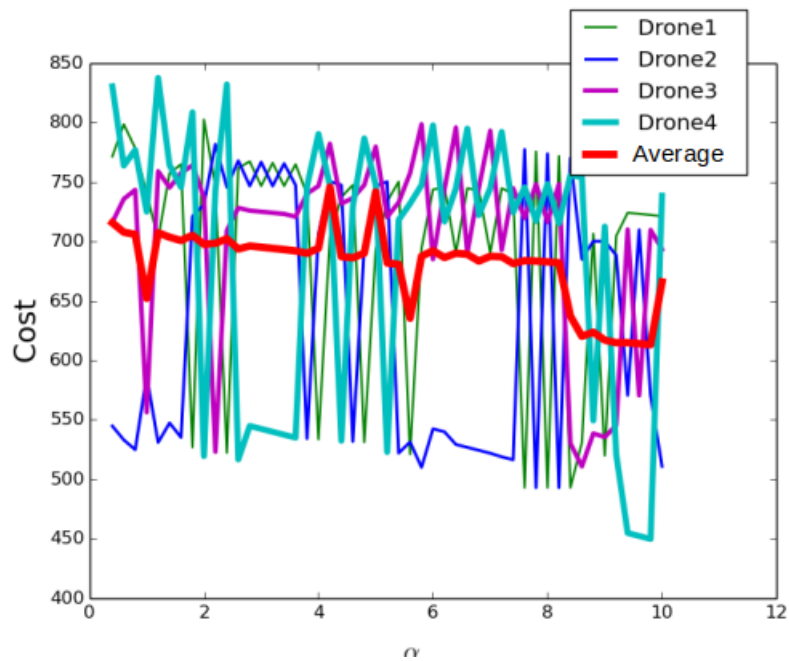


Figure 5.13: Variation of the cost with respect to the overdue time (α).

Figure 5.15 shows the impact of parameter (γ) is incremented by 0.5, at each of 20 consecutive runs. The value of the cost corresponding to each UAV is stochastic, and the average is stochastically decreasing.

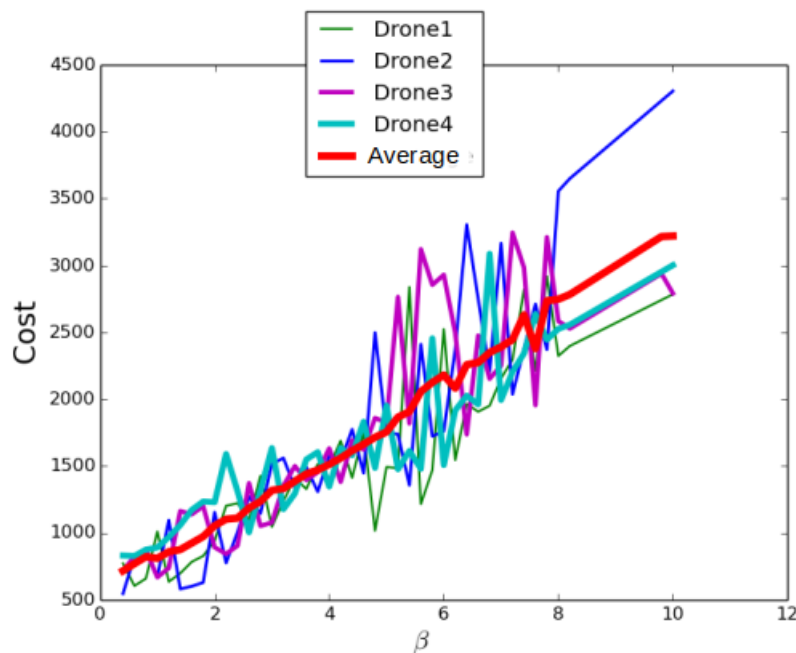


Figure 5.14: Variation of the cost with respect to the delay (β).

The next case to consider is where the latency delay β is the one to change as in Figure 5.23c. The figure shows stochastic values as well but however the average cost is mostly increasing.

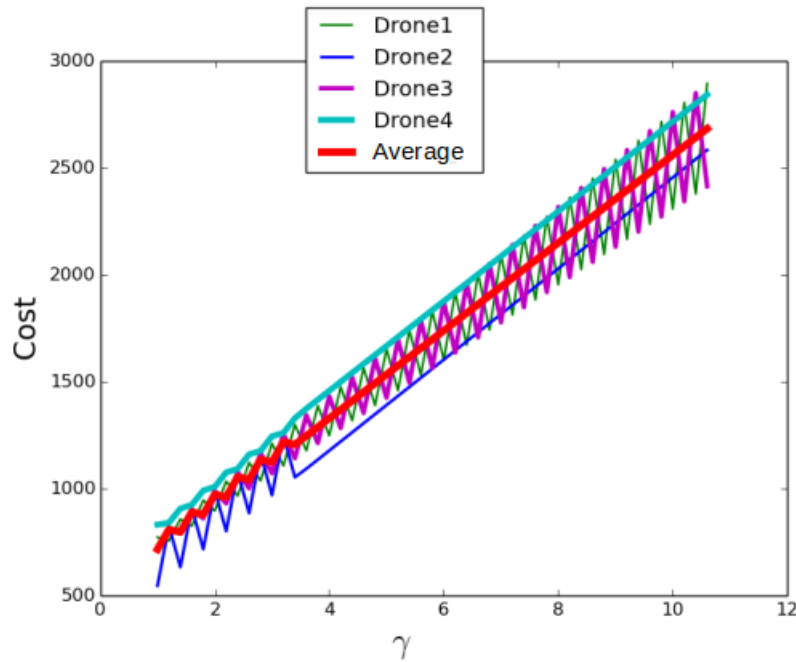


Figure 5.15: Variation of the cost with respect to the data collection rate.

Figure 5.15 shows the impact of varying the data collection rate γ on the variation of the cost over 20 runs. It shows that Drone4 mostly corresponds to the highest cost, and once $\gamma > 3.8$, Drone4 and Drone2 are constantly increasing their respective costs, whereas the costs of the other drones are periodically increasing and decreasing.

5.3.4 Prioritisation analysis. In this subsection, the effect of the delay and overdue constraints is discussed. The network used is as shown in Figure 5.5a where the UAVs *drone1*, *drone2* and *drone3* are initially positioned at the Base Stations $B1$, $B2$ and $B3$, respectively; and all the UAVs are assumed to have the same speed $v=800$ m/min.

Effect of delay constraints on path design

Figure 5.16 and Table 5.5 represent the case where, each sensor node is visited when the arrival of a drone is delayed by no more than 30min.

UAV name	speed (m/min)	Source	Returning path
Drone1	800	B1	[12, B4]
Drone2	800	B2	[14, 13, B3]
Drone3	800	B3	[16, B3]

Table 5.5: Data delivery when UAVs may be delayed by no more than 30 min.

Table 5.5 shows that Drone1 delivers to Base Station $B4$ via node 12, Drone2 to Base Station $B3$ via node 14 and then 13 and Drone3 to Base station $B3$ via node 16.

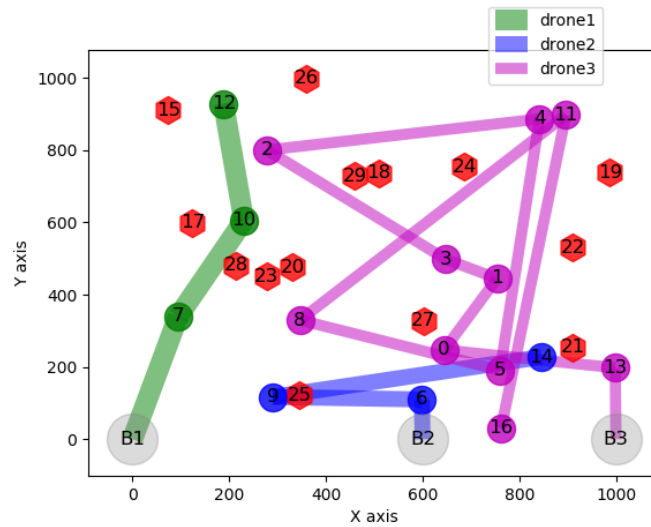


Figure 5.16: Paths details (waiting=30).

Figure 5.16 shows that 14 sensor nodes could not be visited (see red hexagon shaped nodes). The UAV *drone3* visited most of the nodes, while the other UAVs could visit only 3 sensors each.

Table 5.6 shows the data delivery paths when UAVs may be delayed by no more than 60 min, and Figure 5.17 shows the data collection path with this setting.

Table 5.6: Data delivery when UAVs may be late for no more than 60 min.

UAV name	speed (m/min)	Source	Returning path
Drone1	800	B1	[17, B4]
Drone2	800	B2	[21, 13, B3]
Drone3	800	B3	[22, 19, B5]

By changing the lateness threshold to 60 min, Table 5.6 shows that that the delivery paths changed, and the delivery is done as the last column of the table shows.

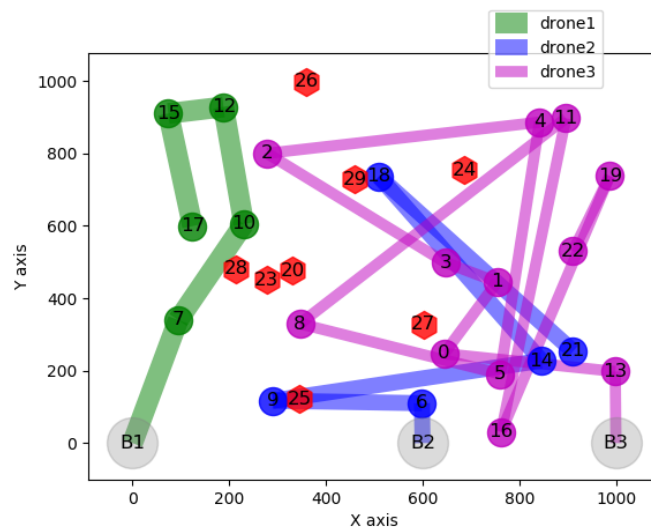


Figure 5.17: Paths details (waiting=60).

Comparing with Figure 5.16, Figure 5.17 reveals that *drone3* does not change its path, but the other UAVs extend their paths to two more sensors nodes each. This results in only 10 sensor nodes being missed.

Table 5.7 and Figure 5.18 show the data delivery and collection paths respectively, when UAVs may be delayed for a very long time. Comparing this case with the previous two cases, Table 5.7 shows that the

Table 5.7: Data delivery when UAVs may be late indefinitely.

UAV name	speed (m/min)	Source	Returning path
Drone1	800	B1	[28, 7, B1]
Drone2	800	B2	[26, 12, B4]
Drone3	800	B3	[29, 15, B4]

delivery paths once again changed.

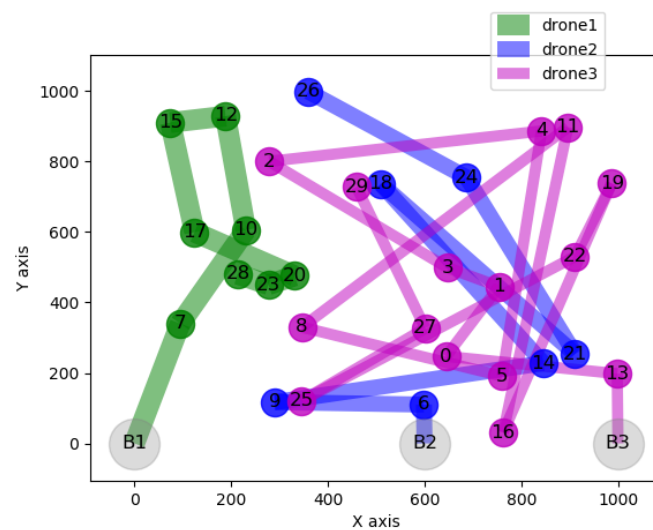


Figure 5.18: Paths details (waiting= ∞).

Figure 5.18 shows that all nodes are visited and the path of each drone have been extended, to cover more nodes.

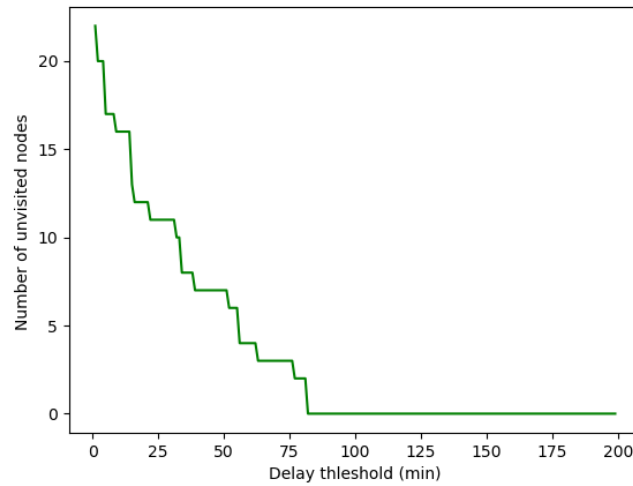


Figure 5.19: Number of unvisited nodes.

Figure 5.19 reveals the changes in the number of missed nodes as a function of the delay threshold. The figure shows that as the delay threshold increases, the number of unvisited nodes remains constant or decreases, until it converges to zero. This shows that, allowing a longer delay increases the chance of all nodes being visited.

Effect of overdue constraints on path design

In this section, the effect of the waiting constraints are studied. Here, sensors can only be visited after some fixed time, called the waiting threshold.



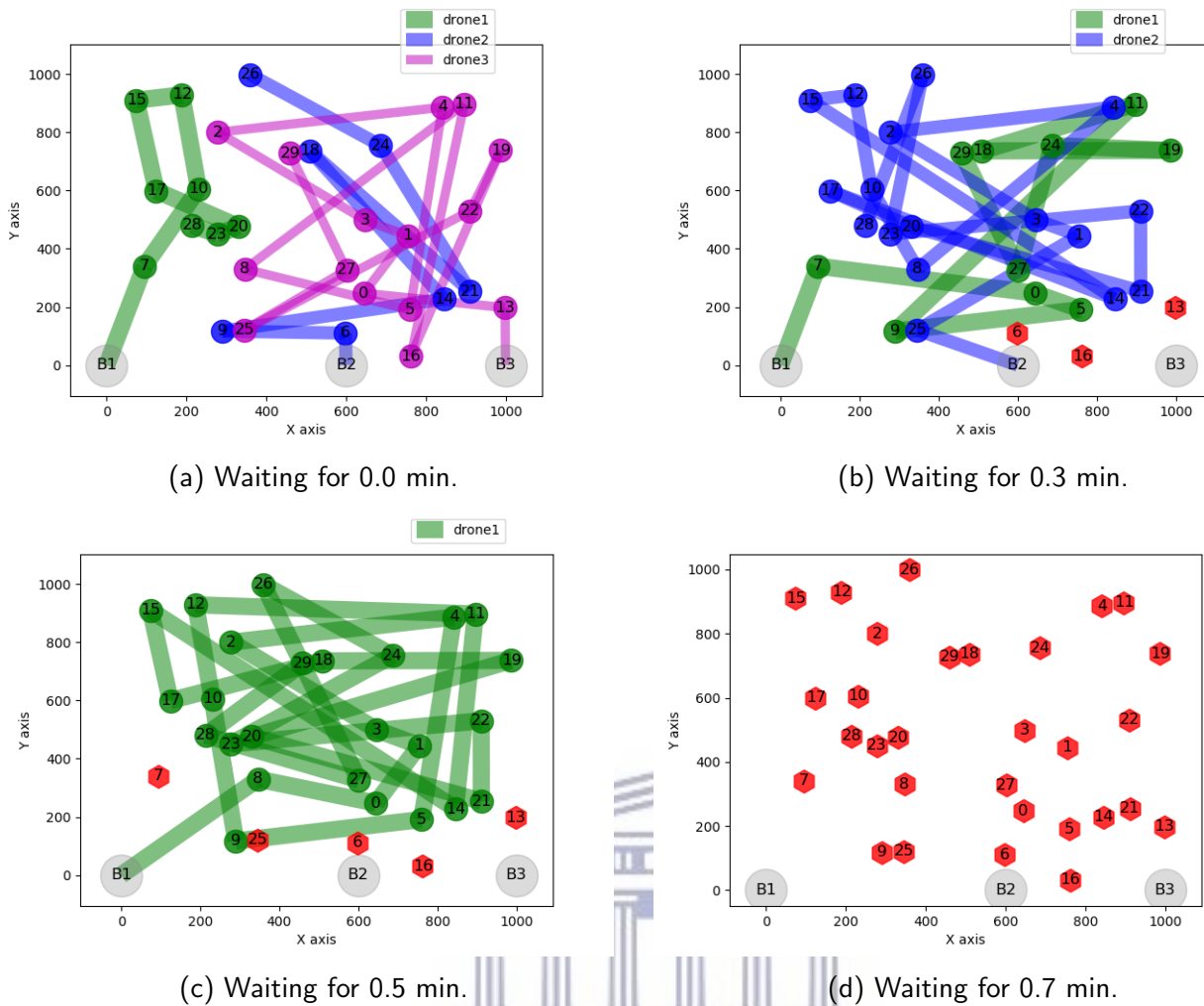


Figure 5.20: Visitation constrained by the waiting time.

Figure 5.20 shows how paths corresponding to different thresholds are generated. Figure 5.20a considers the case where there is no waiting limitation (waiting time is zero) and clearly all sensors are visited. When the waiting time is set to 0.3 minutes (18 seconds) Figure 5.20b shows that three nodes are not visited and the UAV *drone3* could not visit any sensor. Setting the waiting threshold to 0.5 min, only UAV *drone1* could visit and 5 sensors were missed. Setting the waiting time to 0.7 min, no nodes could be visited. This is because it takes a specific time for UAVs to travel between sensors which depends on the distance and speed. If the distance is small and the UAV can arrive earlier than the waiting threshold, the visitation is impossible.

Table 5.8, shows deliveries corresponding to visitations shown in Figure 5.20. Note that if a UAV does not visit any sensor, it remains to its initial position.

Figure 5.21 shows the number of missed sensors as the overdue threshold is increased.

UAV name	speed (m/min)	Source	Returning path
Drone1	800	B1	[28, 7, B1]
Drone2	800	B2	[26, 12, B4]
Drone3	800	B3	[29, 15, B4]

(a) Data delivery when the waiting threshold is 0 min. min.

UAV name	speed (m/min)	Source	Returning path
Drone1	800	B1	[29, 12, B4]
Drone2	800	B2	[B2]
Drone3	800	B3	[B3]

(c) Data delivery when the waiting threshold is 0.5 min.

UAV name	speed (m/min)	Source	Returning path
Drone1	800	B1	[29, 12, B4]
Drone2	800	B2	[28, 7, B1]
Drone3	800	B3	[B3]

(b) Data delivery when the waiting threshold is 0.3 min.

UAV name	speed (m/min)	Source	Returning path
Drone1	800	B1	[B1]
Drone2	800	B2	[B2]
Drone3	800	B3	[B3]

(d) Data delivery when the waiting threshold is 0.7 min.

Table 5.8: Restricted data delivery.

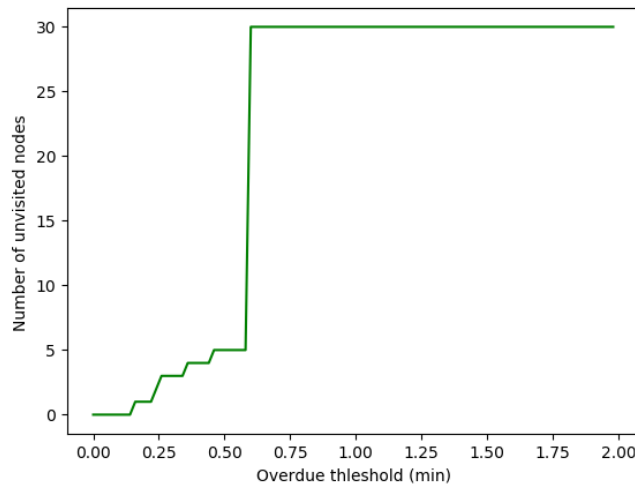


Figure 5.21: Number of unvisited nodes.

Figure 5.21 reveals that the number of missed nodes increases as the waiting threshold increases and converges to the total number of sensors.

Persistent visitation analysis

In this subsection, the trend of paths while UAVs persistently visit sensors is studied. Persistent visitation is done by re-setting the initial base stations for the next visit, to the destination of the previous visitation.

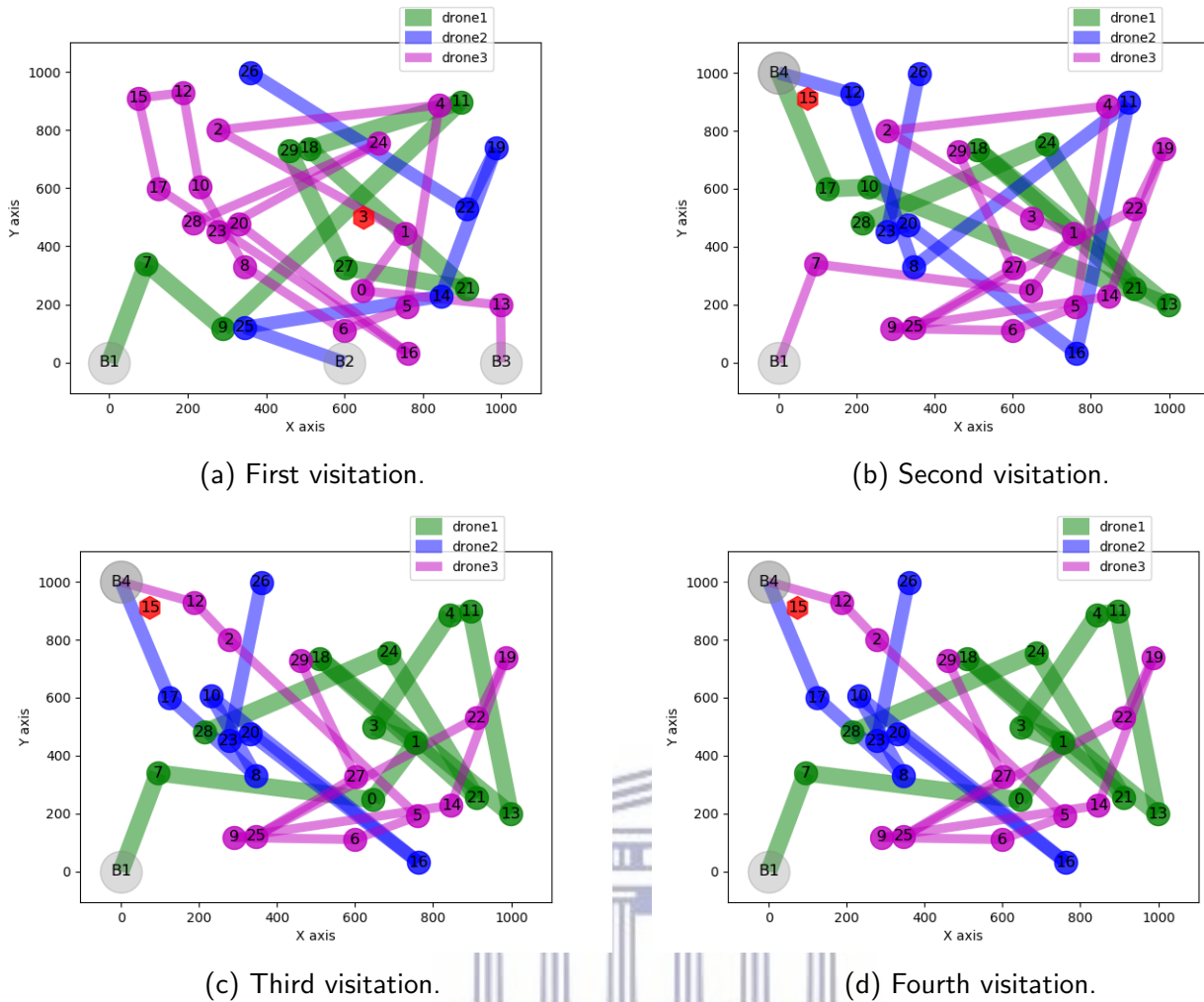


Figure 5.22: Consecutive visitations.

Figure 5.22 shows four consecutive visitations and deliveries when the waiting threshold is set to 12 seconds. Figure 5.22a shows that only node 3 has not been visited. Figure 5.22b shows that a new node (node 15) has not been visited and all visitation paths changes. Figure 5.22c shows that paths keep on changing and unvisited node remains the same as in the previous visitation.

Note that Figure 5.22d is exactly the same as Figure 5.22c. This shows that all next paths will be the same as Figure 5.22c and hence paths generation may converge to specific paths. This is a special case where the initial positions of the UAVs become the same as their optimal destinations.

Constraint free visitations is now considered and the study of the pattern of generated paths is done.

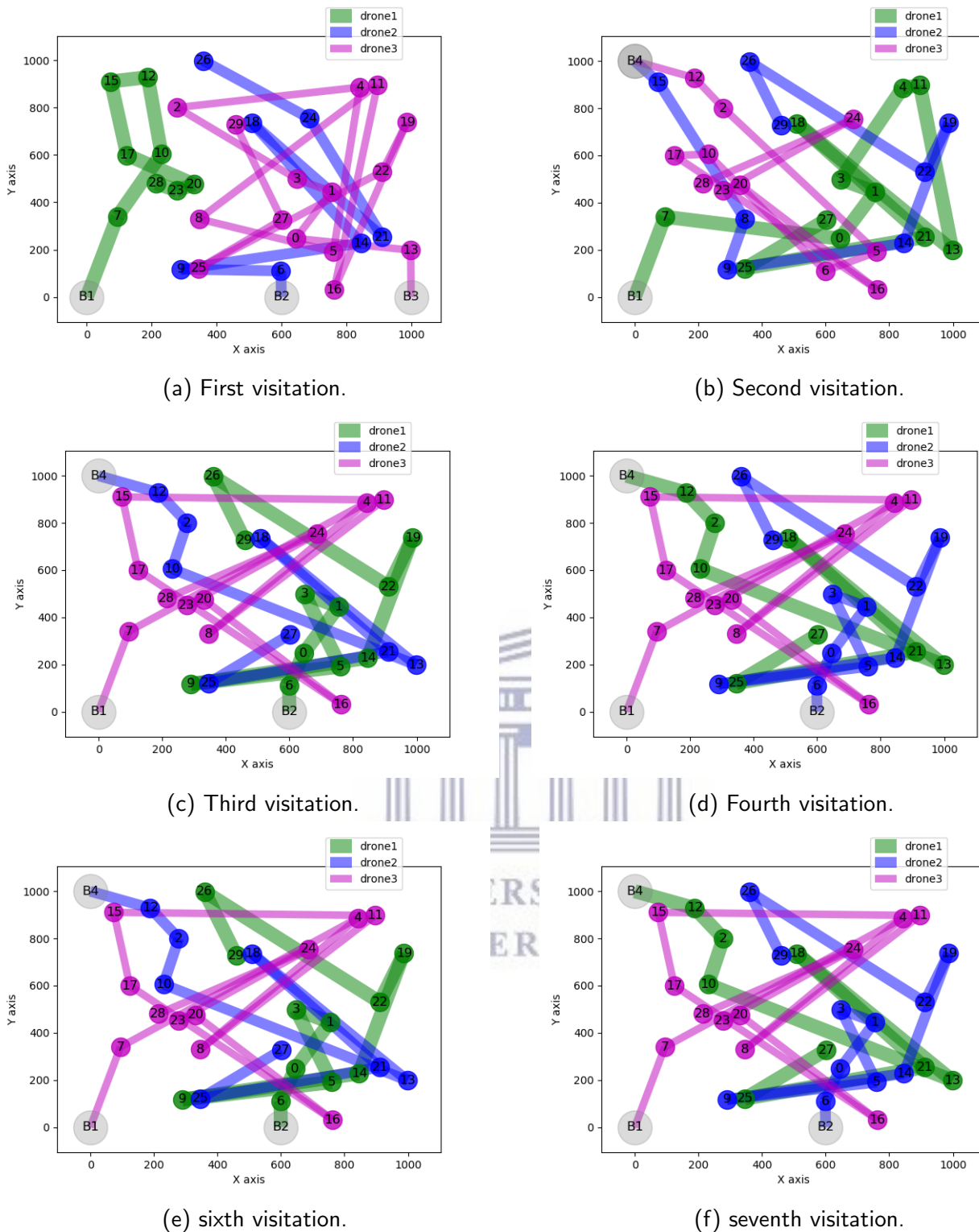


Figure 5.23: Unconstrained visitations.

Figure 5.23 shows the first 6 consecutive unconstrained visitations. Paths change and from the third visitation, paths generation becomes periodic: third visitation is the same as the fifth, and the fourth visitation is the same as the sixth. This shows that after each subsequent pair of visitations the paths generation remains the same. This shows that from each base station, each UAV has a single optimal destination.

5.4 Conclusion

In this chapter, a model for data muling, concentrating on the minimization of revisit costs, for a team of UAVs, has been provided. A mathematical formulation of the model was presented and the underlying problem shown to be NP-hard. A heuristic solution to the problem was thus provided and its performance evaluated through simulation experiments.

Simulation results have revealed a variety of path distribution patterns under different experimental settings and the impact of these settings on path length fairness and related energy costs. Furthermore, these simulations show how consecutive generations become periodic after a small number of first visitations. While this chapter has presented the basis of a data muling model aiming at supplementing traditional traffic engineering techniques used in sink networks, several network and traffic aspects related to the proposed data muling model still need to be investigated. These include the design of efficient communication models that consider the outdoor characteristics of drone-to-sink communication as suggested in [123]. Taking advantage of the emerging white space frequency bands as discussed in [124] to achieve drone-to-sink and drone-to-drone communication is another direction for further work. Furthermore the energy efficiency model needs to be studied to allow the prioritization of less used sensors.



6. Conclusion and future work

6.1 Conclusion

In this thesis, models of hybrid (ground/airborne) networks consisting of a team of UAVs and a ground sensor network have been proposed for the purpose of data muling. This proposed system focuses on four major perspectives, namely: topology optimization and engineering as it relates to single UAVs with multiple targets; topology and traffic optimization for multiple UAVs with single target; scheduling and task allocation for multiple UAVs with single targets and data mulling using multiple UAVs across multiple targets. Models were formulated and analysed for each component. The inter-play between each aspect was addressed to build the system.

A clustering scheme that allows data gathering from select ground sensors (gateways) as proposed. This clustering scheme allowed efficient collection using UAVs as the UAVs only had to visit a small number of sensors to retrieve data. This clustering problem was mathematically defined and has been solved in three main steps.

The popular K-means clustering algorithm was reviewed and found to be best suited for uniformly distributed dense networks. Though a variant exists that optimizes the number of K clusters. These approaches are however not suitable for general networks as a number of nodes were often left out. This work therefore proposed a clustering scheme for general networks which ensures connectivity of all cluster members to corresponding cluster heads. The proposed clustering technique was applied to UAVs clustering especially in instances where the distance between nodes are considerable large.

A routing scheme was proposed which uses a single gateway to facilitate data gathering while continuously minimizes the interference on nodes during message transmission. This scheme minimized not only the interference on nodes but also the distance a message has to travel to the gateway. The routing algorithm was formalized using Z-notation, and its correctness was verified using simulation-based analysis which showed that the proposed algorithm outperformed existing schemes in the same category. On the other hand, considering systems with only one gateway, a UAVs scheduling model was proposed to persistently and periodically visit the gateway referred to as a target. The problem was formulated mathematically and shown to be an NP-hard problem. Three heuristics were proposed, compared and shown to be consistently efficient. Simultaneously, the issue of collision was addressed by relaxing the most consistent of the three heuristics, while also considering the effects collisions had on delay.

Finally, a unifying model was proposed which assumed the existence of a clustered network where data are gathered at cluster heads; a team of UAVs then collect the data and deliver them to base stations for further processing. This UAVs paths planning problem was mathematically formulated and also proven to be NP-hard. A heuristic algorithm was thus proposed and analyzed to solve it. With the effects of varied speed and delay parameters on UAVs paths selection taken into consideration.

6.2 Future Work

This thesis has been proposed as part of the Internet-of-Things in Motion, a project that targets data muling/ferrying using a team of UAVs working in a coalesced manner such as in [125, 126], or independently based on a competitive model as suggested in [127], or a collaborative model as proposed by [128, 129].

The project also targets the sensor network engineering to support UAVs in the data transport processes.

The integration of the proposed networking engineering models will enhance service differentiation in complex sensor networking scenarios with mixed devices as suggested in [130] by balancing sensor roles and UAV proximity is an avenue for future work.

This thesis proposed a clustering model which is also currently being integrated into the smart parking model presented in [131] with service differentiation for ground sensor networks as suggested earlier. The work proposed here, can also be used in the future to complement the work done in [132] as network engineering that considers a hierarchical topology as opposed to the flat topology suggested earlier with the expectation of reducing OPEX and CAPEX. Supporting food security through drought mitigation as suggested in [114, 115] is another technique that, future research work can benefit from the network engineering principles proposed in this paper is by using UAVs as airborne cameras and data mules capable of ferrying agricultural data from fields to processing centres, where machine learning algorithms are applied to improve precision agriculture. Distance-based relaxation was used in the clustering model as a way of mitigating issues related to the energy inefficiency and orphan node issues of the heuristic clustering algorithm. The redistribution of cluster members to achieve a more balanced network is another relaxation technique that can be applied to the two clustering algorithms studied in this paper for energy efficiency and the avoidance of orphan nodes. A combination of distance awareness and cluster members' redistribution is a third technique that can also be applied to the two clustering algorithms. The design and implementation of these techniques is another avenue for future research work.

This thesis further proposed a UAV-aware routing model (Chapter 3) where a team of UAVs collects data from a gateway which have also been optimally selected for both data gathering and collected. The routing model assumes the existence of only one gateway. Model incorporating multi-gateways could be considered in future works. The future work could also include the UAVs communication where, once the data are collected, they will be forwarded to other UAVs for efficient transportation. This will adjust the transportation energy versus the communication energy spent by UAVs.

Chapter 4 proposed a periodic visitation model by many UAVs. A period r has been assumed in formulating the scheduling model. The model could be extended by integrating the data routing mechanisms to compute the optimal visitation period r .

Lastly, this thesis proposed multi-gateway visitation by a team of UAVs where each sensor need to be visited by one UAV (see Chapter 5). However, the model presented in Chapter 3 shows a case where one gateway need to be visited by multiple UAVs. This could steer up future interests in developing related deployment models and also the corresponding assignment schemes where each gateway will be visited by an optimal number of UAVs. On the other hand the volume of data to transport does not translate to the required energy only. It translate to the capacity of transporters (UAVs). Future research included the study on how the volume of data for muling affect the proposed model.

References

- [1] Yi Gu, Qishi Wu, and Nageswara SV Rao. Optimizing cluster heads for energy efficiency in large-scale heterogeneous wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2010, 2010.
- [2] Thomas P Ehrhard. Air force uavs: The secret history. Technical report, Mitchell Inst for Airpower Studies Arlington VA, 2010.
- [3] Robert Zlot and Anthony Stentz. Market-based multirobot coordination for complex tasks. *The International Journal of Robotics Research*, 25(1):73–101, 2006.
- [4] Marc Berhault, He Huang, Pinar Keskinocak, Sven Koenig, Wedad Elmaghraby, Paul Griffin, and Anton Kleywegt. Robot exploration with combinatorial auctions. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1957–1962. IEEE, 2003.
- [5] Amr ME Ahmed, Abhilash Patel, Tom Brown, MyungJoo Ham, Myeong-Wuk Jang, and Gul A Agha. Task assignment for a physical agent team via a dynamic forward/reverse auction mechanism. 2004.
- [6] John Tisdale, ZuWhan Kim, and J Karl Hedrick. Autonomous uav path planning and estimation. *Robotics & Automation Magazine, IEEE*, 16(2):35–42, 2009.
- [7] Roberto Henriques, Fernando Bação, and Victor Lobo. Uav path planning based on event density detection. In *Advanced Geographic Information Systems & Web Services, 2009. GEOWS'09. International Conference on*, pages 112–116. IEEE, 2009.
- [8] Anawat Pongpunwattana and Rolf Rysdyk. Real-time planning for multiple autonomous vehicles in dynamic uncertain environments. *Journal of Aerospace Computing, Information, and Communication*, 1(12):580–604, 2004.
- [9] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun, and Håkan Younes. Coordination for multi-robot exploration and mapping. In *AAAI/IAAI*, pages 852–858, 2000.
- [10] Xu Chu Ding, A Rahmani, and M Egerstedt. Optimal multi-uav convoy protection. In *Robot Communication and Coordination, 2009. ROBOCOMM'09. Second International Conference on*, pages 1–6. IEEE, 2009.
- [11] Nikhil Nigam and Ilan Kroo. Persistent surveillance using multiple unmanned air vehicles. In *Aerospace Conference, 2008 IEEE*, pages 1–14. IEEE, 2008.
- [12] Juan Carlos Rubio, Juris Vagners, and Rolf Rysdyk. Adaptive path planning for autonomous uav oceanic search missions. In *AIAA 1st Intelligent Systems Technical Conference*, pages 20–22, 2004.
- [13] Christopher T Cunningham and Randy S Roberts. An adaptive path planning algorithm for cooperating unmanned air vehicles. In *ICRA*, pages 3981–3986, 2001.
- [14] Zhu Han, KJ Liu, et al. Smart deployment/movement of unmanned air vehicle to improve connectivity in manet. In *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, volume 1, pages 252–257. IEEE, 2006.

- [15] Farhan Mohammed, Ahmed Idries, Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar. Uavs for smart cities: Opportunities and challenges. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 267–273. IEEE, 2014.
- [16] Laszlo Techy, Craig Woolsey, David G Schmale III, et al. Path planning for efficient uav coordination in aerobiological sampling missions. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 2814–2819. IEEE, 2008.
- [17] Mr Daniel Odido and Diana Madara. Emerging technologies: use of unmanned aerial systems in the realisation of vision 2030 goals in the counties. *International Journal of Applied*, 3(8), 2013.
- [18] Enno Littman. Micro aerial vehicles-an eads perspective. Technical report, EUROPEAN AERONAUTIC DEFENCE AND SPACE CO ULMN (GERMANY), 2003.
- [19] Julianne Pepitone. Domino's tests drone pizza delivery. *CNNMoney*, June, 4, 2013.
- [20] Drone delivers flowers to couples in valentine's day stunt. <https://www.campaignlive.com/article/drone-delivers-flowers-couples-valentines-day-stunt/1333490>. Accessed: 2018-04-30.
- [21] Evan Ackerman and Eliza Strickland. Medical delivery drones take flight in east africa. *IEEE Spectrum*, 55(1):34–35, 2018.
- [22] Vincent Tournadre, M Pierrot-Deseilligny, and Paul-Henri Faure. Uav photogrammetry to monitor dykes-calibration and comparison to terrestrial lidar. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(3):143, 2014.
- [23] Drones join war on rhino poachers in south africa. <https://www.telegraph.co.uk/news/worldnews/africaandindian-ocean/africa/10444441/drones-join-war-on-rhino-poachers-in-south-africa.html>. Accessed: 2018-04-30.
- [24] James Mugg, Zoe Hawkins, and John Coyne. *Australian border security and unmanned maritime vehicles*. Australian Strategic Policy Institute, 2016.
- [25] Flavio Araripe dOliveira, Francisco Cristovão Lourenço de Melo, and Tessaleno Campos Devezas. High-altitude platforms present situation and technology trends. *Journal of Aerospace Technology and Management*, 8(3):249–262, 2016.
- [26] DOCTOR INGENIERO DE TELECOMUNICACION. Market-based distributed task allocation methodologies applied to multi-robot exploration.
- [27] John Bellingham, Michael Tillerson, Arthur Richards, and Jonathan P How. Multi-task allocation and path planning for cooperating uavs. In *Cooperative control: models, applications and algorithms*, pages 23–41. Springer, 2003.
- [28] Arthur Richards and Jonathan How. Decentralized model predictive control of cooperating uavs. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 4, pages 4286–4291. IEEE, 2004.
- [29] Klas Andersson, Issac Kaminer, Kevin Jones, Vladimir Dobrokhodov, and Deok-Jin Lee. Cooperating uavs using thermal lift to extend endurance. In *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference*, page 2043, 2009.
- [30] Mehdi Alighanbari and Jonathan P How. Cooperative task assignment of unmanned aerial vehicles in adversarial environments. In *American Control Conference, 2005. Proceedings of the 2005*, pages 4661–4666. IEEE, 2005.

- [31] Tal Shima and Steven Rasmussen. *UAV cooperative decision and control: challenges and practical approaches*. SIAM, 2009.
- [32] Ivan Maza, Konstantin Kondak, Markus Bernard, and Aníbal Ollero. Multi-uav cooperation and control for load transportation and deployment. In *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*, pages 417–449. Springer, 2009.
- [33] Markus Bernard, Konstantin Kondak, Ivan Maza, and Anibal Ollero. Autonomous transportation and deployment with aerial robots for search and rescue missions. *Journal of Field Robotics*, 28(6):914–931, 2011.
- [34] Jonathan Las Fargeas, Pierre Kabamba, and Anouck Girard. Cooperative surveillance and pursuit using unmanned aerial vehicles and unattended ground sensors. *Sensors*, 15(1):1365–1388, 2015.
- [35] J Las Fargeas, P Kabamba, and A Girard. Optimal configuration of alarm sensors for monitoring mobile ergodic markov phenomena on arbitrary graphs. 2015.
- [36] Lisa L Smith, Ganesh K Venayagamoorthy, and Phillip G Holloway. Obstacle avoidance in collective robotic search using particle swarm optimization. In *Swarm Intelligence Symposium*. Institute of Electrical and Electronics Engineers IEEE, 2006.
- [37] BBVL Deepak and Dayal R Parhi. Pso based path planner of an autonomous mobile robot. *Central European Journal of Computer Science*, 2(2):152–168, 2012.
- [38] Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, 1992.
- [39] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the travelling salesman problem. *BioSystems*, 43(2):73–81, 1997.
- [40] Harvey M Salkin and Cornelis A De Kluyver. The knapsack problem: a survey. *Naval Research Logistics Quarterly*, 22(1):127–144, 1975.
- [41] Alain Billionnet and Frédéric Calmels. Linear programming for the 0–1 quadratic knapsack problem. *European Journal of Operational Research*, 92(2):310–325, 1996.
- [42] Peter L Hammer and David J Rader. Efficient methods for solving quadratic 0-1 knapsack problems. *Infor-Information Systems and Operational Research*, 35(3):170–182, 1997.
- [43] Philippe Michelon and Louis Veilleux. Lagrangean methods for the 0–1 quadratic knapsack problem. *European Journal of Operational Research*, 92(2):326–341, 1996.
- [44] David Pisinger. The quadratic knapsack problema survey. *Discrete applied mathematics*, 155(5):623–648, 2007.
- [45] Arnaud Fréville. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research*, 155(1):1–21, 2004.
- [46] Eugene L Lawler. The quadratic assignment problem. *Management science*, 9(4):586–599, 1963.
- [47] Michael Z Spivey and Warren B Powell. The dynamic assignment problem. *Transportation Science*, 38(4):399–419, 2004.
- [48] Dirk G Cattrysse and Luk N Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260–272, 1992.

- [49] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.
- [50] J Mike Spivey. *The Z notation: a reference manual*. Prentice Hall International (UK) Ltd., 1992.
- [51] Graeme Smith. *The Object-Z specification language*, volume 101. Citeseer, 2000.
- [52] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
- [53] Marshall L Fisher and Ramchandran Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.
- [54] Paolo Toth and Daniele Vigo. An overview of vehicle routing problems. In *The vehicle routing problem*, pages 1–26. Society for Industrial and Applied Mathematics, 2001.
- [55] Li-Chun Wang, Chung-Wei Wang, and Chuan-Ming Liu. Optimal number of clusters in dense wireless sensor networks: a cross-layer approach. *Vehicular Technology, IEEE Transactions on*, 58(2):966–976, 2009.
- [56] Enrique J Duarte-Melo and Mingyan Liu. Energy efficiency of many-to-one communications in wireless networks. In *Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on*, volume 1, pages I–615. IEEE, 2002.
- [57] Geng Chen, Fabian Garcia Nocetti, Julio Solano Gonzalez, and Ivan Stojmenovic. Connectivity based k-hop clustering in wireless networks. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 2450–2459. IEEE, 2002.
- [58] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on*, pages 10–pp. IEEE, 2000.
- [59] Haiming Yang and Biplab Sikdar. Optimal cluster head selection in the leach architecture. In *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE International*, pages 93–100. IEEE, 2007.
- [60] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.
- [61] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, 2002.
- [62] Paul S Bradley and Usama M Fayyad. Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99. Citeseer, 1998.
- [63] Hongyuan Zha, Xiaofeng He, Chris Ding, Ming Gu, and Horst D Simon. Spectral relaxation for k-means clustering. In *Advances in neural information processing systems*, pages 1057–1064, 2001.
- [64] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.
- [65] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. An efficient k-means clustering algorithm. 1997.
- [66] Koichi Kise, Akinori Sato, and Motoi Iwata. Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding*, 70(3):370–382, 1998.

- [67] Yi Lu, Shiyong Lu, Farshad Fotouhi, Youping Deng, and Susan J Brown. Fgka: A fast genetic k-means clustering algorithm. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 622–623. ACM, 2004.
- [68] Siddheswar Ray and Rose H Turi. Determination of number of clusters in k-means clustering and application in colour image segmentation. In *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques*, pages 137–143. Calcutta, India, 1999.
- [69] L Lucchesez and SK Mitray. Color image segmentation: A state-of-the-art survey. *Proceedings of the Indian National Science Academy (INSA-A)*, 67(2):207–221, 2001.
- [70] Raihana Ferdous et al. An efficient k-means algorithm integrated with jaccard distance measure for document clustering. In *Internet, 2009. AH-ICI 2009. First Asian Himalayas International Conference on*, pages 1–6. IEEE, 2009.
- [71] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.
- [72] Chunhua Zang and Shouhong Zang. Mobility prediction clustering algorithm for uav networking. In *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*, pages 1158–1161. IEEE, 2011.
- [73] Nanxiang Shi and Xiling Luo. A novel cluster-based location-aided routing protocol for uav fleet networks. *International Journal of Digital Content Technology and its Applications*, 6(18):376, 2012.
- [74] Huseyin Okcu and Mujdat Soy Turk. Distributed clustering approach for uav integrated wireless sensor networks. *International Journal of Ad Hoc and Ubiquitous Computing*, 15(1-3):106–120, 2014.
- [75] Edison Pignaton De Freitas, Tales Heimfarth, Ivayr Farah Netto, Carlos Eduardo Lino, Carlos Eduardo Pereira, Armando Morado Ferreira, Flávio Rech Wagner, and Tony Larsson. Uav relay network to support wsn connectivity. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on*, pages 309–314. IEEE, 2010.
- [76] Marco AM Marinho, Edison Pignaton De Freitas, João Paulo C Lustosa da Costa, André Lima F de Almeida, and Rafael Timóteo de Sousa. Using cooperative mimo techniques and uav relay networks to support connectivity in sparse wireless sensor networks. In *Computing, Management and Telecommunications (ComManTel), 2013 International Conference on*, pages 49–54. IEEE, 2013.
- [77] Sanghamitra Bandyopadhyay, Chris Giannella, Ujjwal Maulik, Hillol Kargupta, Kun Liu, and Souptik Datta. Clustering distributed data streams in peer-to-peer environments. *Information Sciences*, 176(14):1952–1985, 2006.
- [78] Souptik Datta, Kanishka Bhaduri, Chris Giannella, Ran Wolff, and Hillol Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, 10(4):18–26, 2006.
- [79] Souptik Datta, Chris Giannella, and Hillol Kargupta. Approximate distributed k-means clustering over a peer-to-peer network. *IEEE Transactions on Knowledge and Data Engineering*, 21(10):1372–1388, 2009.
- [80] Emmanuel Tuyishimire, B Antoine Bagula, and Adiel Ismail. Optimal clustering for efficient data muling in the internet-of-things in motion. In *International Symposium on Ubiquitous Networking*, pages 359–371. Springer, 2018.

- [81] Emmanuel Tuyishimire. Internet of things: least interference beaconing algorithms, 2014.
- [82] Franz Aurenhammer, Rolf Klein, Der-Tsai Lee, and Rolf Klein. *Voronoi diagrams and Delaunay triangulations*, volume 8. World Scientific, 2013.
- [83] S Skiena. Dijkstra's algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pages 225–227, 1990.
- [84] the free encyclopedia Wikipedia. Dominating set.
- [85] Roger Coud. Radio mobile.
- [86] Antoine Bagula, Marco Zennaro, Gordon Inggs, Simon Scott, and David Gascon. Ubiquitous sensor networking for development (usn4d): An application to pollution monitoring. *Sensors*, 12(1):391–414, 2012.
- [87] A Fehnker, RJ van Glabbeek, P Höfner, A Mclver, M Portmann, and WL Tan. A process algebra for wireless mesh networks used for modelling, verifying and analysing aadv. Technical report, Tech. Rep. 5513, NICTA, 2013.
- [88] RW Duke and Gordon Rose. *Formal object-oriented specification using Object-Z*. Macmillan, 2000.
- [89] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, Sukun Kim, Philip Levis, and Alec Woo. The collection tree protocol (ctp). *TinyOS TEP*, 123:2, 2006.
- [90] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14. ACM, 2009.
- [91] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *ACM SIGOPS operating systems review*, volume 34, pages 93–104. ACM, 2000.
- [92] Elmouatezbillah Karbab Antoine Bagula, Djamel Djenouri. Ubiquitous sensor network management: The least interference beaconing model. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. IEEE PIMRC, 2013.
- [93] Antoine Bigomokero Bagula, Djamel Djenouri, and Elmouatezbillah Karbab. On the relevance of using interference and service differentiation routing in the internet-of-things. In *Internet of Things, Smart Spaces, and Next Generation Networking*, pages 25–35. Springer, 2013.
- [94] Jan Blumenthal, Ralf Grossmann, Frank Golatowski, and Dirk Timmermann. Weighted centroid localization in zigbee-based sensor networks. In *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, pages 1–6. IEEE, 2007.
- [95] Mauro Franceschelli and Andrea Gasparri. Decentralized centroid estimation for multi-agent systems in absence of any common reference frame. In *American Control Conference, 2009. ACC'09.*, pages 512–517. IEEE, 2009.
- [96] Krzysztof Pawlikowski, H-DJ Jeong, and J-SR Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications magazine*, 40(1):132–139, 2002.
- [97] Marco Zennaro and Antoine B Bagula. Design of a flexible and robust gateway to collect sensor data in intermittent power environments. *International Journal of Sensor Networks*, 8(3-4):172–181, 2010.

- [98] Marco Zennaro, Antoine Bagula, David Gascon, and Alberto Bielsa Noveleta. Planning and deploying long distance wireless sensor networks: The integration of simulation and experimentation. In *International Conference on Ad-Hoc Networks and Wireless*, pages 191–204. Springer, 2010.
- [99] Jonathan Las Fargeas, Baro Hyun, Pierre Kabamba, and Anouck Girard. Persistent visitation with fuel constraints. *Procedia-Social and Behavioral Sciences*, 54:1037–1046, 2012.
- [100] Jonathan Las Fargeas, Baro Hyun, Pierre Kabamba, and Antoine Girard. Persistent visitation under revisit constraints. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 952–957. IEEE, 2013.
- [101] Nikhil Nigam, Stefan Bieniawski, Ilan Kroo, and John Vian. Control of multiple uavs for persistent surveillance: algorithm and flight test results. *Control Systems Technology, IEEE Transactions on*, 20(5):1236–1251, 2012.
- [102] Noa Agmon, Sarit Kraus, Gal Kaminka, et al. Multi-robot perimeter patrol in adversarial settings. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2339–2345. IEEE, 2008.
- [103] Yehuda Elmaliach, Asaf Shiloni, and Gal A Kaminka. A realistic model of frequency-based multi-robot polyline patrolling. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, pages 63–70. International Foundation for Autonomous Agents and Multiagent Systems, 2008.
- [104] Sara Hedberg. Robots cleaning up hazardous waste—the use of robots for hazardous waste cleanup is gaining acceptance as new technologies are invented, tested, and proven. *AI expert*, 10(5):20–25, 1995.
- [105] J Colegrave and A Branch. A case study of autonomous household vacuum cleaner. *AIAA/NASA CIRFFSS*, 107, 1994.
- [106] Mazda Ahmadi and Peter Stone. A multi-robot system for continuous area sweeping tasks. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1724–1729. IEEE, 2006.
- [107] Yehuda Elmaliach, Noa Agmon, and Gal A Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3-4):293–320, 2009.
- [108] Joel G Manathara, PB Sujit, and Randal W Beard. Multiple uav coalitions for a search and prosecute mission. *Journal of Intelligent & Robotic Systems*, 62(1):125–158, 2011.
- [109] Christelle Guéret, Christian Prins, and Marc Sevaux. Applications of optimization with xpress-mp. *contract*, page 00034, 1999.
- [110] Antoine B Bagula. Modelling and implementation of qos in wireless sensor networks: a multiconstrained traffic engineering model. *EURASIP Journal on Wireless Communications and Networking*, 2010(1):468737, 2010.
- [111] Antoine B Bagula and Anthony E Krzesinski. Traffic engineering label switched paths in ip networks using a pre-planned flow optimization model. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*, pages 70–77. IEEE, 2001.
- [112] Antoine Bagula, Lorenzo Castelli, and Marco Zennaro. On the design of smart parking networks in the smart cities: An optimal sensor placement model. *Sensors*, 15(7):15443–15467, 2015.

- [113] Antoine Bagula, Marco Zennaro, Gordon Inggs, Simon Scott, and David Gascon. Ubiquitous sensor networking for development (usn4d): An application to pollution monitoring. *Sensors ISSN 1424-8220*, 12(7):391–414, 2012.
- [114] Muthoni Masinde and Antoine Bagula. A framework for predicting droughts in developing countries using sensor networks and mobile phones. In *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 390–393. ACM, 2010.
- [115] Muthoni Masinde, Antoine Bagula, and T. Nzioka Mthama. The role of icts in downscaling and up-scaling integrated weather forecasts for farmers in sub-saharan africa. In *In proceedings of ICTD'12*, pages 122–129. ACM, 2012.
- [116] M. Mandava, C. Lubamba, A. Ismail, H. Bagula, and A. Bagula. Cyber-healthcare for public healthcare in the developing world. In *Proceedings of the 2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 14–19. IEE, 2016.
- [117] Antoine Bagula, Claude Lubamba, Munyaradzi Mandava, Adiel Ismail, Herman Bagula, , Marco Zennaro, and Ermmano Pietrosevoli. Cloud based patient prioritization as service in public health care. In *Proceedings of the ITU Kaleidoscope 2016, Bangkok, Thailand, 14-16 November*. IEE, 2016.
- [118] Emmanuel Tuyishimire, Ismail Adiel, Slim Rekhis, B Antoine Bagula, and Noureddine Boudriga. Internet-of-things in motion: A cooperative data muling model under revisit constraints. In *proceedings of the 13th IEEE International Conference on Ubiquitous Intelligence and Computing, presented on 2016 July, 18-21 Toulouse, France*.
- [119] Antoine Bagula, Emmanuel Tuyishimire, Jason Wadepoel, Noureddine Boudriga, and Slim Rekhis. Internet-of-things in motion: A cooperative data muling model for public safety. In *proceedings of the 13th IEEE International Conference on Ubiquitous Intelligence and Computing, presented on 2016 July, 18-21 Toulouse, France*.
- [120] Michael A Bender, Sándor P Fekete, Alexander Kröller, Vincenzo Liberatore, Joseph SB Mitchell, Valentin Polishchuk, and Jukka Suomela. The minimum backlog problem. *Theoretical Computer Science*, 605:51–61, 2015.
- [121] Gui Citovsky, Jie Gao, Joseph SB Mitchell, and Jiemin Zeng. Exact and approximation algorithms for data mule scheduling in a sensor network. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 57–70. Springer, 2015.
- [122] Anandabrata Pal and Nasir Memon. The evolution of file carving. *Signal Processing Magazine, IEEE*, 26(2):59–71, 2009.
- [123] Marco Zennaro, Hervé Ntareme, and Antoine Bagula. Experimental evaluation of temporal and energy characteristics of an outdoor sensor network. In *Proceedings of the International Conference on Mobile Technology, Applications, and Systems, Article 99*. ACM, 2008.
- [124] Timothy X Brown, Ermmano Pietrosevoli, Marco Zennaro, Antoine Bagula, Hope Mauwa, and Sindiso M Nleya. A survey of tv white space measurements in e-infrastructure and e-services for developing countries., 2015, pp 164-172. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Volume 147*, pages 164–172. EAI, 2015.

- [125] Adiel Ismail, Bigomokero Bagula, and Emmanuel Tuyishimire. Internet-of-things in motion: A uav coalition model for remote sensing in smart cities. *Sensors*, 18(7):2184, 2018.
- [126] Adiel Ismail, Emmanuel Tuyishimire, and Antoine Bagula. Generating dubins path for fixed wing uavs in search missions. In *International Symposium on Ubiquitous Networking*. Springer, 2018.
- [127] Antoine Bagula, Emmanuel Tuyishimire, Jason Wadepoel, Noureddine Boudriga, and Slim Rekhis. Internet-of-things in motion: A cooperative data muling model for public safety. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pages 17–24. IEEE, 2016.
- [128] Emmanuel Tuyishimire, Antoine Bagula, Slim Rekhis, and Noureddine Boudriga. Cooperative data muling from ground sensors to base stations using uavs. In *Computers and Communications (ISCC), 2017 IEEE Symposium on*, pages 35–41. IEEE, 2017.
- [129] Emmanuel Tuyishimire, Ismail Adiel, Slim Rekhis, B Antoine Bagula, and Noureddine Boudriga. Internet of things in motion: A cooperative data muling model under revisit constraints. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pages 1123–1130. IEEE, 2016.
- [130] Antoine Bagula, Ademola Philip Abidoye, and Guy-Alain Lusilao Zodi. Service-aware clustering: An energy-efficient model for the internet-of-things. *Sensors*, 16(1):9, 2015.
- [131] Antoine Bagula, Lorenzo Castelli, and Marco Zennaro. On the design of smart parking networks in the smart cities: An optimal sensor placement model. *Sensors*, 15(7):15443–15467, 2015.
- [132] Luca Chiaraviglio, Nicola Blefari-Melazzi, William Liu, Jairo A Gutiérrez, Jaap van de Beek, Robert Birke, Lydia Chen, Filip Idzikowski, Daniel Kilper, Paolo Monti, et al. Bringing 5g into rural and low-income areas: Is it feasible? *IEEE Communications Standards Magazine*, 1(3):50–57, 2017.