KING'S College LONDON

University of London

6CCS3PRJ Final Year
Flying Base Stations in 5G and Beyond - Mobile Wireless Networks

Final Project Report

Author: Rakan Zabian

Supervisor: Vasilis Friderikos

Student ID: 1741706

April 23, 2020

Abstract

The abstract is a very brief summary of the report's contents. It should be about half-a-page long. Somebody unfamiliar with your project should have a good idea of what your work is about by reading the abstract alone.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Rakan Zabian

April 23, 2020

Acknowledgements

It is usual to thank those individuals who have provided particularly useful assistance, technical or otherwise, during
your project. Your supervisor will obviously be pleased to be acknowledged as he or she will have invested quite a lot of time overseeing your progress.

Contents

Chapter 1

Introduction

1.0.1 Project Motivation

The complementation of different technologies harvested by the Fourth Industrial Revolution is key to leaping towards the fully automated world of tomorrow. For such reasons, mobile data networks have become an integral and underlying structure of the world. The initial evolution of wireless networks was primarily a virtually infinite network of computers, key to the advancements of the 20th century and the restructuring of its economies [1]. The evolution of the internet brought by the advent of Third Generation wireless networks (3G) initiated the mobile internet, introducing the connectivity of devices. Fourth Generation wireless networks (4G) satisfied the growing thirst of the world to use the internet even more dynamically. This advent witnessed the emergence of the Internet of Things (IoT), which is currently connecting a number of devices over six times times larger than the global human population. Every advancement introduces newer network complexities and challenges such as higher data rate requirements and the limitation of available frequency spectrum. The existing wireless systems will not be able to handle the exponential increases in mobile broadband data accelerated by the increased Machine-to-Machine (M2M) communications [8]. The much anticipated deployment of the Fifth Generation of wireless networks (5G) is expected to accommodate the increased data rates and requirements by providing larger scale connectivity. This will be done by enhancing current network technologies to improve the networks, and by introducing new technologies that will evolve a new network with unique standards. However, as previously mentioned, every advancement in the networks introduces more complex challenges. 5G networks introduce a new spectrum range that offers extreme capacity and speed, but this comes at the expense of coverage. The high frequency waves used in the upper bound of the spec-

trum range that 5G employs travel at lower wavelengths, introducing the need for densification of the networks. More Terrestrial Macro Base Stations (MBSs) will need to be deployed to provide connectivity, with small cells dominating the urban areas of cities to complement the Macro Cell layer. However, due to geographical constraints and cost limitations, many regions will not be able to smoothly deploy the higher band spectrum of 5G networks. Recently, the deployment of Unmanned Aerial Vehicles (UAVs) in wireless networks has received significant attention and research to approach such constraints. The flexibility of UAV networks can overcome geographical constraints and cost limitations if deployed optimally. Developments such as Google's Project Loon and projects aiming to enable wide-scale communications through UAV employment by ATT and Qualcomm are motivating further research into the deployment of UAVs to act as Flying Base Stations (FBSs) [5] . FBSs can augment the terrestrial 5G network by enhancing operations in a plethora of varying use cases such as data collection for IoTs, information dissemination, and machine-type communications. The FBSs can be used to support high communication demand areas such as large events or highly dense regions. In such use case scenarios, the UAV assisted 5G networks can provide seamlessly ubiquitous connectivity at any location, potentially well beyond the coverage area of a single terrestrial MBS. This will improve the network efficiency and flexibility and thus ease the growth in high data demand.

1.0.2 Report Structure

This project aims to discuss some 5G technologies and to highlight the essence of UAV assisted 5G networks by expanding on the value introduced by FBS deployment. Research addressing problems associated with the networks such as energy limitation, optimal 3D deployment and path planning will be discussed. Finally, the project will delve into the trajectory optimization of FBSs and the optimal cell edge placement of terrestrial MBSs. The structure of the remainder of the report is as follows.

Chapter 2 will give a background on UAV assisted 5G networks by exploring the current advancements in 5G networks and IoT technologies, and by discussing the value brought by integrating FBSs into the networks. Some of the researched problems associated with FBS deployment will then be discussed. The chapter will end with a literature review on the research of UAV assisted 5G networks.

Chapter 3 will delve into the primary problem associated with the Trajectory Optimization

of FBSs, by analysing the problem and its solutions. Algorithms that intend to optimize FBS trajectory will then be proposed and explained, followed by an analysis of the simulations that implement the proposed algorithms.

Chapter 4 will evaluate the problems through numerical investigation. The simulation results will be presented with an in depth analysis on the insights obtained from the results.

Chapter 5 will summarize the project and discuss its limitations. The potential for future works on the researched topics will be discussed.

Chapter 2

Background

2.1 5G Network Technologies

As previously discussed, existing wireless communication networks will not accommodate the increase in broadband data and are currently encountering fundamental challenges. Examples of such challenges are: higher data rate and Quality of Service (QoS) requirements, energy efficiency and excellent end-to-end performance and user coverage in overcrowded areas and hotspots whilst maintaining extremely low latency and high bandwidth. The deployment of 5G networks aims to address such challenges by introducing multiple advancements to the network and implementing new technologies to evolve new radio networks. This will primarily be done by introducing the 5G New Radio, which is the radio technology that is being developed to support the 5G technologies that will solve the problems mentioned previously. With the New Radio implementation, the next generation networks will accommodate the growing data rates. The networks are expected to attain a mobile data volume per unit area that is 1,000 times higher than current networks. Over 10-100 times the number of current connected devices is expected to be accommodated by 5G networks. This is envisioned as the evolving cellular networks will adopt a multi-tier architecture of macrocells, a variety of small cells, relays and a composite of other heterogeneous networks to serve specified user types, equipment and areas. The radio aims to maximize energy and spectrum efficiency in the process [8]. Some of the key New Radio technologies will be discussed below.

2.1.1 New Radio Spectrum

With spectrum consisting of frequencies below 2.5GHz, considered Low-band spectrum, being very limited in bandwidth due to the overcrowding and congestion of signals in the low-band range, 5G considers a different solution. 5G solutions aim to offer a higher range of frequencies lying in the Mid to High band spectrum range. This will offer higher bandwidth as the upper bounds of the spectrum have little to no congestion, offering a high available of frequencies. What was considered High-band spectrum for 4G is Mid-band for 5G. The standard is now higher with the New Radio so that the requirements can be met and the current network problems can be solved. The adoption of Mid to High band spectrum for 5G will introduce extreme capacity and speed using millimeter waves. Millimeter waves have very high frequencies and low wavelengths, which means that the capacity and throughput will increase at the expense of coverage. Densification of networks in urban areas and the employment of heterogeneous networks will aim to solve these problems. However, an advantage of low wavelengths is smaller antenna sizes, which allows for the adoption of Massive Multiple-Input-Multiple-Output systems which will.

2.1.2 Multiple Input Multiple Output Systems and Beamforming

With 5G spectrum range, the deployment of massive MIMO systems is achievable. It is extremely challenging for Massive MIMO systems such as Full Dimension Massive MIMO to support Low-band spectrum ranges. This is because the lower band frequencies require larger antennas to accommodate the large wavelength. Antenna size significantly decreases with Mid to High band frequencies, which 5G New Radio aims to take full advantage of. Using Massive MIMO support grids in the network systems will enable signals to be aimed in three dimensions. The technology can also enhance network coverage and capacity unlike prior LTE features. With Massive MIMO systems comes an increased Beamforming ability, which can help drive user-specific beams precisely into space. This can even be user equipment (UE) directed with more sophisticated digital beamforming. The enablement of beamforming by Massive MIMO can also help reduce overall interference by precisely steering user-specific beams at cell edge users. The higher Signal to Noise Ratios (SNRs) caused by the complementation of the aforementioned technologies will also allow for higher signal modulation schemes such as 256 QAM (Quadrature amplitude modulation), which will largely increase throughput.

2.1.3 Software-Defined Networking

For network operators to enhance network scalability and flexibility so that specific demands of industries or UE are met, key technologies are required to be implemented with the 5G advancement. The primary technology that will enable such flexibility in monitoring and scaling the network, similar to that of cloud computing services, is Software Defined Networking. The technology will allow providers to move away from hardware-defined systems that are strictly rigid to more adaptable software based systems. Network Function Virtualization (NFV) is a complementary technology to SDN that will virtualize networks to make them easier to upgrade to more advanced 5G services. Networks will respond better to real-time demands with this implementation of NFV [intel 5g cloudification]. NFV does not depend on SDN, however, some models do. [vasilis] notes that the fail-fast model, a model enabling the rapid scaling of successfully deployed ideas or the quick discarding of failed ideas is fully dependant on SDN. For such reasons, the virtualization of 5G greatly depends on SDN. The cloud-like networks can then employ excelling network qualities such as network slicing, a much anticipated technology of 5G. With this technology, networks can self-contain virtualized network slices to serve a diverse range of specific needs. With the arrival of SDN, the networks will attain flexibility, self-configuration and service of critical communications at heights previously unknown by networks.

2.1.4 Use Cases

5G promises the fruition of ideas only currently imagined by the general public, and researched vigorously in the research and development sectors. Such use cases are the advancement of self-driving cars due to the extremely low latency network which will allow almost instantaneous communications that will be considered safe enough to deploy the CAVs. The ultra-low latency will also enable the implementation of remote and semi-autonomous robot that are currently reforming the manufacturing industry, and will potentially be advanced and further developed to be used for even more critical industries such as healthcare and remote surgery. This will re-form healthcare and industrial automation to heights previously unattainable. This integration with IoT is only possible as the billions of IoT devices that are to be cloud connected depend on the highly capable network that 5G technology brings. Smart capabilities will be enabled for almost all devices and systems, increasing efficiency, safety and convenience.

2.2 UAV Network Deployment

As noted in the previous section, coverage is primarily the crucial problem with 5G networks, requiring the densification of urban areas with heterogeneous networks and the deployment of more closely packed terrestrial MBSs. However, this is not cost-effective and can be more complex as terrestrial network replanning will be required. The issue can be overcome by integrating UAVs into the network infrastructure as FBSs. With the primary requirement of 5G networks being data rate, FBSs are seen as excellent provisions of continuous support of the UEs in networks. The deployment of UAVs as FBSs has provided solutions for CAPEX/OPEX (CAPital EXpenditure/OPerational EXpenditure) [22]. The dynamic implementation of UAV has allowed the envisioning of multiple use case scenarios of UAVs inside network infrastructure and in other areas such as public safety and military services. However, each application of UAVs requires a study of UAV types and the environment the UAV will be deployed in to evaluate what UAV would be most suitable for a specific mission.

Classification

A classification of UAVs can be based on the altitudes at which the UAVs fly. High Altitude Platform (HAP) UAVs fly at altitudes about 17km, and can be deployed for long term projects, potentially lasting for multiple months due to the UAV's long endurance. The UAVs are quasi-stationary and can provide wide coverage. Low Altitude Platform (LAP) UAVs can typically fly for up to several hours at an altitude of up to few kilometers. The UAVs are highly cost-effective and quick to deploy and possess quick mobility.

Type classifications assess the qualities of the UAVs. Fixed-wing UAVs, such as small aircrafts, travel at high speed for several hours and can carry high payload. However, the UAVs cannot hover which is a concern for specific use cases such as FBS deployment, as FBSs are required to hover over Ground Nodes (GNs) to provide higher QoS for longer time durations. The hovering issue associated with fixed-wing UAVs is solved with Rotary-wing UAVs, which can hover, such as drones. However, these UAVs travel at lower speeds, for a duration that is typically less than an hour, and the UAVs are also energy limited [15]. It is for such reasons that UAV types must be assessed so that missions can choose UAVs suitable for the tailored mission.

Use Cases

A plethora of use cases with UAVs is envisioned due to their efficient and flexible deployment. For example, consider a scenario in which a natural disaster has damaged a terrestrial network in a region, and victims of the region no longer have connectivity. Public safety communications are then highly compromised due to the disaster. The deployment of UAVs in this case would be a highly effective alternative to the damaged or highly overcrowded terrestrial MBS. The UAV can be deployed as FBSs to temporarily enable reliable communications with high QoS regardless of the terrestrial damage. The flexibility of the UAVs is also a reason for their effective public safety deployment as the UAV positions can be easily adjusted according to the GN distribution, effectively establishing public safety communications [14].

The increased adoption of IoT devices has been a key challenge of wireless networks as previously discussed. The networks are slowly evolving into a heterogeneous IoT environment consisting of a variety of devices such as sensors and mobile devices. Sensors in particular have become increasingly deployed as they are key to a plethora of use cases such as efficient garbage disposal, transportation, energy management, and smart city infrastructure management. Such applications of IoT require seamlessly ubiquitous connectivity to maintain connectivity as devices are largely battery and energy limited and thus fail to transmit data over large instances. The IoT platform can be assessed through four layers: devices, connectivity and networks, platform and marketplace, and applications and use cases. Some IoT devices require ultra low latency, and almost all require energy efficiency and reliability whilst maintaining high-speed uplink communications. The wide range of applications for the devices makes them also deployable in non serviced areas such as deserts. All the aforementioned issues are not as critical in some conventional cellular network use cases. Thus, the deployment of UAVs as FBSs is also applicable in the IoT environment scenario to provide reliable uplink communications for the IoT devices [13]. The dynamic placement of the FBSs can be utilized to minimize the energy and power consumption of the IoT devices by lowering altitudes or hovering above specified locations.

The UAV use case that will be explored in this project is the FBS application for coverage and capacity enhancement. In this regard, UAVs can be highly effective in reforming the current terrestrial network into a more dynamic, self-configurating and flexible network. Consider 5G networks, as mentioned previously the approach to tackle the coverage problem is building

a heterogeneous network consisting of the terrestrial MBS, small cells, femtocells, and picocells for connectivity users [8]. The deployment of these devices will improve coverage and increase capacity, however, as noted by [3] the network restructuring due to the device deployment is complex and costly to a large extent. FBSs can be a more intelligent deployment in this regard due to their flexibility, agility, mobility, and adaptive altitude in providing service. The FBSs can act as pivots that reliably and efficiently relay information between the MBSs and the cells while supporting direct connectivity between GNs. The FBSs can also rapidly serve large surges in demand in some hotspots. This has been explored by ATT and Verizon in using drones as internet providers for large gatherings like the Superbowl. The FBSs can also intelligently relocate to areas that are not well served by a terrestrial MBS based on the demand. Another key problem FBSs can tackle is the low LoS service provided by 5G MBSs with barriers dispersed in the cell. The FBS can provide LoS communcation as the aerial nature of the FBSs allows for them to minimize the effects of blockage, which can be used to provide millimeter wave connectivity in some cases. The FBSs can also improve energy efficiency of networks by reducing unnecessary consumption of energy which can when a MBS is only serving few users. The FBS can then take over from the MBS by serving those users itself, allowing a fraction of the MBS to become inactive, saving energy and minimizing terrestrial interference in the network [21]. To conclude, it is evident that UAVs have a large potential in being dynamically employed for a variety of missions. The FBS have shown to be potentially crucial to terrestrial MBS sites as the FBSs can provide cost-effective, fast, and seamlessly reliable connectivity to poorly covered areas or regions suffering from an overcrowding or high demand surge [16].

Problems Associated with UAV Deployment

To harvest the benefits of and efficiently deploy FBSs, several challenges that will be discussed in this section must be firstly overcome. Regulations are challenging to overcome as drone usage is constrained by government institutions worldwide such as the Civil Aviation Authority in the United Kingdom. Operational Altitudes are often limited, which may be unfavorable for environments that require high altitudes for higher LoS connectivity. The different types of drones must be investigated to assess how they could be used to optimally provide service. For example, solar powered unmanned vehicles could be more cost effective than electrically powered drones [21]. The number of drones to be assigned per BS is also a demanding investigation as multi-tier drone architecture with different types of drones bringing a diversity of essential strengths to the network may be a solution to the limited capability and endurance of a single

drone [].

Research commonly addresses energy efficient deployment of the FBSs. This must be achieved to provide service effectively as FBSs, particularly rotary-wing UAVs, are highly limited in duration and must be optimized in all aspects. Operational altitude is also a challenging investigation when addressing the 3D placement of drones [15]. The optimal number of drones to be assigned per MBS is another problem that arises when considering a multi-tier drone architecture with different types of drones [23]. Another key problem then arises, which is the path planning of a single FBS to serve all GNs before returning to the MBS, or simultaneously deploying multiple FBSs to serve a region. This project will address two problems: trajectory optimization of FBSs and clustering of GNs. The first problem will be addressed by planning an optimal path for the FBS, and the second will be addressed by optimally locating the cell edge of a terrestrial MBS.

2.3 Literature Review

Paper [24] addressed the minimization of total Rotary-wing UAV consumption for wireless communication. The paper addressed different types of energy consumption such as propulsion, hovering, and communication related energy consumption. The paper succeeded in deriving a propulsion power consumption model which helped minimize the total UAV energy consumption. Traveling Salesman Problem (TSP)and convex optimization techniques were effectively used to demonstrate optimal hovering locations for the UAVs while still satisfying target GN communication throughput. An algorithm was proposed to solve the modelled optimization problems from the power consumption model, and the algorithm produced results which minimized energy consumption for the UAVs. The algorithm significantly outperformed other approaches discussed in the paper such as the geometric center fly hover approach or the GN hover approach. Paper [20] weighed multiple approaches to solving the TSP for the trajectory optimization of UAVs, and the results displayed the superiority of the 2-opt NN algorithm. This motivated further investigation into using the algorithm in this project. The algorithm outperformed the PSO and GA heuristics both in terms of computational time and distance in some experimental results displayed in the paper. The paper also discussed the K-means clustering algorithm and other clustering approaches to be integrated with a TSP solver. This was also a source of motivation to implement some of the techniques in this project to optimize trajectory. The TSP was also largely investigated in papers [11] [13] [9]. Some of the papers provided results of algorithm experiments which were used to evaluate algorithms in

this project. It was unknown whether the 2-opt algorithm could run exponentially in a special case of point distribution, until paper [6] addressed this issue by presenting such distributions. The below table summarizes some significant topics in FBS optimization by highlighting the papers addressing each problem.

Chapter 3

System Model

3.1 FBS Trajectory Optimization: Introducing the Traveling Salesman Problem

For the FBS to serve all user nodes in the region appointed to it, the route of travel must begin from the MBS and visit every node only once before returning to the MBS. The cost of travel for the FBS, which is the total distance covered when traveling, must be minimized to optimize the FBS' trajectory. The lower the cost, the more efficient the trip will be with regards to the limited power, energy and speed of the FBS. This is an example of the Traveling Salesman Problem (TSP), an NP-hard (Non-deterministic polynomial time hardness) problem in combinatorial optimization. The TSP will be described in the following section, and the NP-hardness will be further explained in section 3.1.3.

3.1.1 Description

The TSP asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that passes through each city only once before returning to the origin city?". The problem can be modelled as a weighted graph where the vertices represent the cities and the weights of the edges represent the distances between each pair of cities. The TSP would be to find the shortest Hamiltonian Cycle of the graph [18].
The TSP can either be symmetric (STSP) or asymmetric (ATSP). Let the set of Vertices on the graph be $V = \{v_1, v_2, v_3..., v_n\}$, the set of edges be $E = \{(a, b)|a, b \in V\}$, and the associated cost between two vertices be c_{ab}. The cost is equivalent to the distance between the two cities. In the STSP, the distance between two cities is equivalent in both opposite directions. Consequently,

the STSP can be modelled as an undirected graph where $d_{ab} = d_{ba}$. In the ATSP, the distance between two cities is different in both opposite directions. Consequently, the ATSP can be modelled as a directed graph where $d_{ab} \neq d_{ba}$ [13]. The STSP will be referred to as the TSP in the future of this project as the FBS deals with the STSP.

3.1.2 Integer Linear Programming Formulation

[13] Presented one of the most cited formulations of the TSP. The model is the Dantzig-Fulkerson-Johnson (DFJ) Integer Linear Programming Formulation of the TSP [4].
As mentioned in the previous section, the set of vertices on the graph is defined by $V = \{v_1, v_2, v_3..., v_n\}$, the set of edges by $E = \{(i,j)|i,j \in V\}$, and the associated cost between two vertices by c_{ij}. The formulation is as follows.

Minimize

$$\sum_{i<j} c_{ij} x_{ij} \tag{3.1}$$

Subject to

$$x_{ij} \in \{0,1\} \qquad (i,j) \in E \tag{3.2}$$

$$\sum_{i<k} x_{ik} + \sum_{k<j} x_{kj} = 2 \qquad (k \in V) \tag{3.3}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1 \qquad (S \subset V, 3 \leq |S| \leq -3) \tag{3.4}$$

(3.1) Is the objective function which captures the total distance covered by the route. It uses the decision variable $x_{ij} \in \{0,1\}$ to highlight whether an edge (i,j) exists or not. The cost of the edge c_{ij} will only be added to the total trip distance if $x_{ij} = 1$ and the edge (i,j) exists. If $x_{ij} = 0$, then there is no such edge and the cost of the edge will not be added to the total trip distance. The function aims to minimize the total trip distance by optimizing the route though the selection of the best edges, subject to constraints (3.2,3.3,3.4).

(3.2) Is the integrality constraint which guarantees that x_{ij} will be either 0 or 1. In other words, the edge (i,j) will either be added to the route or it will be ignored.

(3.3) Is the degree constraint which guarantees that each vertex will be connected directly

to only two other vertices, through two edges. For any vertex $k \in V$ between vertices i and j, there will be only two vertices connected to k as the sum of the decision variables will be 2.

(3.4) Is the subtour elimination constraint, which prevents any subtours from forming. If there exists a set of vertices $S \subset V$ with $|S|$ elements, the total number of edges in the set will have an upper bound $|S| - 1$. In other words, no set of vertices belonging to V will have a closed loop except the complete set V [13].

3.1.3 Computational Complexity

The TSP belongs to the class of NP-hard problems. To further understand this, NP-hardness will be explained.
The set P is the set of all problems for which there exists an algorithm that runs in polynomial time. The set NP is the set of all problems for which an existing solution can be verified in polynomial time. Consider the decision problems D and L. There exists a reduction of problem D to problem L if there exists a function F such that:

1. F can be computed in polynomial time.
2. F maps each input of D to an input of L.
3. For every input f_1 of D, the answer is 'yes' if and only if the answer of $F(f_1)$ of L is 'yes'.

A problem is NP-hard if it can be reduced to from an NP problem. Any problem that is NP-hard but is verifiable in polynomial time is classed as NP-complete. NP-complete is the intersection between NP and NP-hard [12].

There is no known polynomial time solution for the TSP, and the TSP is also not necessarily verifiable in polynomial time. For these reasons, the TSP is at least is as hard as the hardest NP problem and is thus classed as NP-hard. The Euclidean TSP, if input with rational coordinates and a discretized metric, is classed as NP-complete. The Euclidean TSP will be further discussed in the following section.

3.1.4 Special Cases

When the distance between two cities in a TSP is the Euclidean distance between the corresponding points, the TSP is considered to be a Euclidean TSP. The Euclidean TSP is an

example of a metric TSP which requires all distances between points to satisfy the triangle inequality. This means that the direct route between two points is shorter than the route passing through an intermediate point. The trajectory optimization problem faced by the FBS is an example of the Euclidean TSP.

3.2 Methods of Computing a Solution

As previously mentioned, the TSP is an NP-hard optimization problem. The three traditional methods used to attack NP-hard problems are as follows.

1. Formulating exact algorithms, which will output the optimal solution for the problem.
2. Using heuristics, which will output approximate solutions that are sub-optimal.
3. Using exact heuristics for a sub-problem of the problem, a special case of the problem.

3.2.1 Exact Algorithms

With linear programming, optimal solutions can be devised for reasonably small instances. However, since the TSP is NP-hard, finding optimal solutions becomes infeasible as the instances get larger. Exact algorithms will yield excellent quality solutions at the expense of time [18].

Brute-Force Search

The direct solution to the TSP would be to try out all possible permutations of the cities in a brute-force search. After trying all possibilities, the search will locate the shortest possible route. However, since the brute-force search will try out all possible permutations, the running time will be $O(N!)$. This is an inefficient and infeasible approach as the number will grow at a rate that is even greater than an exponential function of n. For only 15 cities, the brute-force search will need to try out over 1×10^{12} permutations. Branch and Bound algorithms have solved large instances of the TSP by pruning the brute-force search space.

Held-Karp (HK) Algorithm

[7] devised an algorithm which applies dynamic programming to solve the TSP. The algorithm solves the TSP in running time $O(2^n \times n^2)$. The HK lower bound is the solution to the linear programming relaxation of the integer linear programming formulation of the TSP. For these reasons, the HK lower bound is commonly used as a standard of measurement of the

performance of TSP heuristics. An iterative version of the HK lower bound proposed by [HK] to run more feasibly for very large instances keeps within 0.01% of the actual HK lower bound, which averages approximately 0.8% below the optimal route distance [18].

3.2.2 Heuristics and Approximation Algorithms

Since exact algorithms become infeasible for growing instances of the TSP, optimality can be compromised for sub-optimal solutions to gain feasibility. That is the primary objective of a heuristic or an approximation algorithm. Heuristics and approximation algorithms settle for near optimal solutions to decrease the growth of exact algorithm complexities from exponential to polynomial. Heuristics cannot always guarantee a worst-case scenario. A theorem in [19] claims that no local search algorithm taking polynomial time per move can guarantee a worst-case ratio 'c' if $P \neq NP$. This ratio is a constant which guarantees that any solution produced using the algorithm will be less than $c \times o$ where o is the optimum. In other words, the heuristic used to solve a problem can sometimes produce the optimal solution, and in other times, with a varying input, can produce the worst possible solution for the same problem. An example of this will be discussed when explaining the Nearest Neighbour (NN) Algorithm later in this section. Heuristics will gain speed and simplicity at the expense of quality. Constructive heuristics build new solutions and stop when a solution is found. The best tour construction algorithm produces a solution within 10-15% of the optimal solution. Improvement heuristics improve current solutions. Unlike heuristics, approximation algorithms will formally guarantee a worst-case ratio 'c' [18].

Sanjeev Arora's Algorithm

The best known approximation algorithm is Sanjeev Arora's Polynomial Time Approximation Scheme (PTAS). The scheme is based on dynamic programming [6]. The algorithm runs in polynomial time finding a solution that has a worst case ratio of $1+\frac{1}{c}$ where $c > 0$ for geometric instances of the TSP. The algorithm runs in $O(n \log_2(n)^{(O(c\sqrt{d}))^{d-1}})$ time where d represents the number of dimensions in the Euclidean space.[S arora] For the common two dimensional TSP, the complexity of the algorithm would be $O(n \log_2(n)^{(O(c)})$.

Nearest Neighbour (NN) Algorithm

The NN algorithm is a constructive heuristic. To understand the NN algorithm, greedy algorithms will be explained. Greedy algorithms are algorithms that make the locally optimal

choice at each step, so that that algorithm can compute the global optimum. They are quick to design and implement and can run quickly, however, their output quality can vary widely depending on the input. The NN algorithm is a greedy algorithm which starts at a point and gradually completes the tour by always moving to the nearest point, adding it to the tour. The algorithm will always find the nearest city by following the steps listed below.

1. Select a random city as the origin and endpoint of the route.
2. Locate the nearest unvisited city and move to it.
3. If there are unvisited cities remaining, repeat step 2.
4. Return to the first city

The NN algorithm will yield a solution that is within an average of 25% above the HK lower bound. However, there exists distributions of points that will cause the NN algorithm to produce the worst route possible. The algorithm has a complexity of $O(n^2)$.

The Greedy Heuristic

The greedy heuristic, also known as the Multi-fragment algorithm is also a constructive heuristic. It progressively forms the route by always picking the shortest edge. This is constrained by the following constraints. Firstly, the edge must not increase the degree of any point to more than 2. Secondly, it must not form a closed loop that does not pass through each point. Thirdly, no edge can be added more than once. The algorithm follows the below steps.

1. Sort all the edges.
2. If the shortest edge does not violate any of the constraints, add it to the tour by selecting it.
3. If the tour is not yet complete, repeat step 2.

The greedy algorithm will yield a solution that is within an average of 15-20% above the HK lower bound. The algorithm has a complexity of $O(n^2 \log_2(n))$.

Christofides' Algorithm

Christofides' algorithm is an approximation algorithm which uses graph theory by combining the Double Minimum Spanning Tree (MST) algorithm with the minimum-weight perfect matching (MWM) of a graph. The Double MST algorithm is as follows.

1. Find and build a MST from the set of all cities
2. Duplicate all edges, which can be done using the Depth First Search (DFS) algorithm, to create a Eulerian graph and construct a Eulerian cycle.
3. Convert the graph to TSP by traversing the cycle and taking shortcuts when a node has been visited by skipping the node to prevent visiting a node more than once.

The Double MST algorithm guarantees a worst case ratio of 2 and a complexity $O(n^2 \log_2(n))$. Christofides' algorithm was successful in improving the Eulerian graph by following an approach different from the duplication of all edges. The previous algorithm produced the graph by doubling every edge of the MST, and thus making all vertex degrees even. This will output a Eulerian graph, however, the graph will have an excessive number of edges, making the TSP conversion more time consuming and less accurate. Christofides' algorithm took a different approach by combining the MST with the MWM of all vertices of the MST with odd degrees. This will produce a better Eulerian graph by adding edges only the odd degree vertices to make them even. The MWM is produced using algorithms of complexity $O(n^3)$. Christofides' algorithm follows the below steps.

1. Find and build a MST from the set of all cities
2. Create a MWM on the set of all vertices which have an odd degree. Add the MWM to the MST.
3. Convert the graph to TSP by traversing the cycle and taking shortcuts when a node has been visited by skipping the node to prevent visiting a node more than once.

Christofides' algorithm succeeded in improving the worst case ratio of the previous algorithm to $\frac{3}{2}$. Christofides' algorithm has a complexity $O(n^3)$. Experiments have shown the algorithm to produce outputs that are about 10% above the HK lower bound [18].

2-opt Algorithm

Local search algorithms are heuristics that gradually move through the search space of candidate solutions, applying local moves to find the local optimum. The algorithms will stop once a solution that is considered globally optimal is found. Local search is considered to be the most successful approach to finding heuristic solutions in global combinatorial optimization. The 2-opt algorithm is a local search iterative improvement heuristic that improves a solution by replacing two edges of a tour with two other edges. This is called a 2-opt move, which more

precisely will replace two current crossing edges $(e_1, e_2), (e_3, e_4)$ with the edges (e_1, e_3),(e_2, e_4). The move is subject to the constraint that the replacement will not create any sub-tours. Therefore, there will only be one possible move when the initial two edges are to be replaced which will be implemented only if the new tour is shorter than the previous tour. The heuristic is said to achieve amazingly good results in terms of approximation ratio and running time when considering "real word" Euclidean instances [6]. The algorithm follows the following procedure.

1. Let S be the current solution, the Hamiltonian cycle of the graph.
2. Select a node i and examine the possibility for 2-opt moves to the edge connecting i to i's successor. Apply if possible and if the cycle length will be decreased.
3. Repeat step 2 for every node i until no possible 2-opt changes exist. The tour is now 2-optimal.

[11] The average number of moves of the 2-opt will vary depending on the initial solution. For example, if the 2-opt improves a greedy algorithm, the average number of moves will be $O(n)$. For random starts, the average will be $O(nlog)$ The initial number of moves is 10 times smaller than that for the improvement of random starts. For Euclidean distances, the 2-opt will yield a solution that is about an average of 5% above the HK lower bound. The only certain bound that can be placed for the cost of finding an improving move is $\Theta(n^2)$ for the 2-opt [9].

K-opt Algorithm

Similarly to the 2-opt, there exists a 3-opt algorithm that will improve on the quality of the 2-opt by replacing three crossing edges rather than two. The 3-opt can be seen as multiple 2-opt moves on a set of three crossing edges. The only certain bound that can be placed for the cost of finding an improving move is $\Theta(n^3)$ for the 3-opt [9]. For Euclidean distances, the 3-opt will yield a solution that is about an average of 3% above the HK lower bound. K-opt follows the same fashion as the 2-opt with K edges. The process can continue with 4-opt and so on, but as K increases, the algorithm running times will increase while the rate of improvement in quality of the algorithms will decrease.

V-opt Algorithm

The V-opt is a generalization of the K-opt, but rather than using a fixed K, the V-opt will grow the set of K's as the search continues. The Lin-Kernighan (LK) algorithm is the best known

algorithm that applies the V-opt method. The LK algorithm decides what is the most suitable K for each iteration. The algorithm follows the below steps.

1. Given a Hamiltonian cycle, or a tour, delete K mutually disjoint edges.
2. Reconnect the remaining fragments so that no sub-tours are formed.
3. There are $2K - 2$ possibilities to reconnect the fragments to form a valid tour. With reasonable K's a brute-force search can be applied to find the re-connection with the lowest cost.

The algorithm will yield a solution that can possibly get within 2% above the HK lower bound. The complexity of the algorithm is approximately $O(n^{2.2})$, making it slower to run than a simple 2-opt implementation.

Simulated Annealing (SA)

The process of SA has been successfully adapted to model physical processes to give approximate solutions to combinatorial optimization problems such as the TSP. SA models the physical process of slowly changing the temperature of a system to reach a ground state. This can be visualized by considering the cooling of a liquid to the freezing point so that a crystalline structure is formed. The temperature must be slowly decreased so that at each step the system state relaxes to the state of minimum energy [11]. When applying this to the TSP, the ground state of energy would be the optimal tour. At every step of temperature variation, random moves will be implemented so that the tour could potentially improve. This will change the solution by either improving it or making it worse, so energy increases are likely to occur. At a low temperature, it is more probable that the solution is closer to the optimum. The algorithm follows the below steps.

1. Compute a Hamiltonian cycle H and choose a temperature T and a repetition factor R.
2. Implement a random modification of H to produce H'. If c(H')¡c(H) then replace H with H'. Otherwise, compute a random number $0 \leq n \leq 1$ and replace H with H' if $n \leq e^{-\frac{c(H')-c(H)}{T}}$. Repeat this step R times.
3. Modify T and R. Repeat steps 2,3 if the stopping criterion is not yet satisfied.
4. Output the best solution that was found.

[11] An implementation of the SA algorithm presented in [9] uses 2-opt moves to find neighbouring solutions, running for $O(n^2)$ time with a large constant of proportionality. Since the algorithm uses 2-opt moves as part of its implementations, SA achieves results that are comparable to the 2-opt results. If the algorithm is allowed to run longer, some results are comparable to LK results.

Tabu Search

Tabu search was developed to improve the process that other algorithms such as SA and some of the nature-mimicking algorithms that will be discussed after Tabu Search. Rather than only allowing for negative gain moves to occur so that the global optimum is searched for, the tabu search will also prevent a solution from being repeated again. Tabu search follows the below steps.

1. Compute a Hamiltonian cycle H and an empty tabu list T.
2. Implement the best move not on T.
3. Update T and repeat steps 2 and 3 until the stopping criterion is satisfied.
4. Output the best solution found.

[8] Most implementations of Tabu Search will take up to $O(n^3)$ time. The quality will be better than 2-opt moves if 2-opt implementations are used, but the 2-opt algorithm is significantly faster when run alone.

Algorithms Mimicking Natural Procedures

Some algorithms such as Genetic Algorithms (GA) and Ant Colony Optimization have been produced by observing the solutions nature produce for complex problems. GAs mimic genetic recombinations, which is the producing of offspring using exchanged genetics. This produces offspring traits that are not existent in either parent.[wiki genetic recom] When applied to the TSP, a genetic algorithm will use multiple tour solutions and mutate them to improve their quality or use them as parents to generate an offspring tour with a lower cost than that of the parent tours. GA's have shown results that surpass LK results, however, running time is an issue.

Ant Colony Optimization mimics the disposal of pheromone trails in ants. When ants search for food, pheromones are disposed to guide other ants on the same trail so that more food can be found. Ant Colony Optimization (ACO) applies this by starting with a group of ants, typically

around 20. The ants are randomly dispersed among the cities and are then asked to move to another city that is unvisited by themselves unless the tour will be completed. Each ant will produce a virtual pheromone trail which will have a strength that is inversely proportional to the tour length. The strength of the trails will be considered by each ant when choosing the next city. The process is repeated until an acceptable tour is found.

3.3 Evaluation of Methods

To compare all presented methods and conclude with one that will be used to set the FBS trajectory, only feasible methods will be considered. Since the TSP faced by the FBS is a practical issue requiring the routing of UAVs, more practical methods must be applied rather than strong theoretical methods. For these reasons, only heuristics will be compared as exact algorithms and [Aroras] are either infeasible or impractical. For the TSP, the GA was found to be a better algorithm than ACO in . However, the GA requires more complex inputs than most traditionally used algorithms for the TSP such as: best values for the chromosome population, the crossover and the mutation probabilities [2]. The SA algorithm also requires multiple inputs such as the temperature and the repetition. SA will yield better results than the 2-opt, comparable to those of the LK algorithm, only if run for a longer time. The FBS trajectory must be found in a quicker, simpler and more practical manner. Therefore, the ACO, Genetic and SA algorithms will not be considered in deciding what algorithm will be used to determine the FBS trajectories.

3.3.1 Comparison of Heuristics

There are multiple ways of finding the best solution from a list of heuristics. If the specific interest is speed, the priority can be given for faster algorithms with compromised quality. If seeking excellent performance, speed must be compromised and better performing algorithms can be considered. To assess the speed and performance of an algorithm the worst case approximation times will give an idea to where an algorithm lies in terms of speed, however, the worst case running time is not enough. Due to the widely varying and uncertain nature of heuristics, the worst case running time may not be a sole indicator of the actual average running time of an algorithm. The same goes for the performance, specific cases can yield widely varying results. For these reasons, multiple sources of data on the actual CPU running times of the algorithms together with their performance provided by experimentation will be considered.

The data will help to verify or modify the conclusions that will be made from the worst case running times and performances listed in (3.2.2).

Algorithm	% excess over HK Lower Bound	Worst case running time
Greedy	15	$O(n^2 \log_2(n))$
Christo	10	$O(n^3)$
2-opt	5	$\Theta(n^2)$
3-opt	3	$\Theta(n^3)$
LK	2	$O(n^{2.2})$
Tabu	$< 5\%$	$O(n^3)$

Table 3.1: Evaluation of Heuristics

As mentioned previously for the FBS trajectory, practical computation time is crucial. However, quality cannot be too compromised as this will then significantly lower the UAV durability due to the increased cost. A conclusion solely based on this table is as follows. No constructive heuristic can be left to run alone as this will achieve a quality of 10% above the HK lower bound at best. Therefore, the NN, Greedy and Christofides' algorithms will not be considered to be solely depended on without improvement. However, one can be used as a starting solution that can be improved upon using an iterative improvement algorithm.

Christofides' algorithm has the highest complexity followed by the NN algorithm. This difference compared to NN's running time must be greatly considered as the chosen constructive heuristic's complexity will add to that of the chosen iterative improvement algorithm. In addition to this, [11] states the following. Firstly, when applying simple iterative improvement algorithms, a reasonable starting solution is recommended as the improvement algorithm is not powerful enough for random starts. Secondly, the best suited start for an iterative improvement algorithm is the NN algorithm as it outputs good pieces of a Hamiltonian cycle with few bad pieces that could be easily improved be repaired. Thirdly, when the 2-opt is implemented to a farthest insertion solution, the results are much worse than when the 2-opt improves a NN solution even though the insertion heuristic yields much better Hamiltonian tours than the NN does [9].

For the above reasons, the NN algorithm will be chosen as the constructive heuristic that will be used to output a starting solution for the FBS trajectory.

The improvement algorithms are difficult to compare for the following reasons. Firstly, as mentioned in , no local search algorithm iterating in polynomial time can guarantee a worst case ratio c. Secondly, due to the first reason, the complexities may be bad indicator of actual running times. Thirdly, the three algorithms perform similarly well and are all not too difficult to implement. The fastest and simplest to implement is the 2-opt, as it will require less moves than the 3-opt, LK and Tabu algorithms due to it having a lower and constant K value. Tabu and 3-opt have significantly slower running times than the 2-opt so they will be eliminated as viable options. However, LK yields significantly improved quality with a closeness of 1% to the HK lower bound, which could possibly be worth the less significant difference in running time with the 2-opt. The 2-opt may be the best choice as it runs quickly and is easier to implement, however, the data that will be presented will be used to make a final conclusion. The below figure was taken from [10].

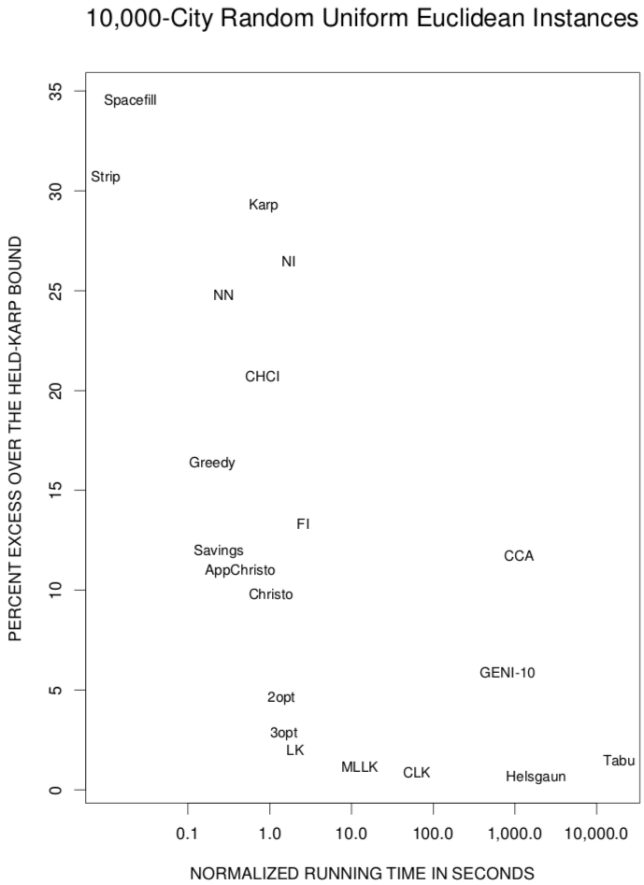

Figure 3.2: Average Trade-offs between Tour Quality and Normalized Running Time

The above data supports the first conclusion derived: no constructive heuristic such as greedy, Christo or NN can be solely depended on. The three are far too distant from the optimum,

especially when compared to the 2-opt, 3-opt or LK. The same goes for eliminating Tabu as it is significantly the longest running algorithm on the graph, so it can definitely be eliminated as a viable option for outputting the FBS trajectory. Based on the graph, 2-opt, 3-opt and LK algorithms are all close candidates once again. Using this graph, LK and 3-opt could potentially surpass the 2-opt as a viable FBS trajectory finder due to the minor difference in running time. However, this graph is the result of runs using 10,000 cities. The FBS simulation that will be discussed in (3.4) will use nodes sizes ranging from 20 to 80 nodes as this is traditional when clustering users. For these reasons, further experiments must be analysed to come to a final conclusion. A final data-set of iterations with city sizes that are closer to the cell node sizes in the simulations will be analysed. The below tables use data from tables in [11]. The results were collected after using the algorithms to solve multiple problem instances listed in the TSPLIB.

Table 3.3: Quality of Candidate Improvement Heuristics as % Excess Over the Optimum

TSP Problem	2-opt	3-opt	LK
lin105	8.42	0.00	0.77
pr107	3.79	2.05	1.53
pr124	2.58	1.15	2.54
pr136	10.71	6.14	0.55
pr144	3.79	0.39	0.56
pr152	2.93	1.85	0.00
u159	14.00	11.49	2.20
rat195	6.46	3.01	1.55
d198	3.85	6.12	0.63
Mean	6.28	3.58	1.15
Standard Deviation	1	2	3

Table 3.4: CPU Times of Candidate Improvement Heuristics in Seconds(s)

TSP Problem	2-opt	3-opt	LK
lin105	0.02	4.10	1.15
pr107	0.01	2.73	0.71
pr124	0.02	3.82	1.03
pr136	0.03	3.76	1.05
pr144	0.02	6.37	1.20
pr152	0.02	3.44	1.03
u159	0.02	6.03	1.46
rat195	0.02	4.10	1.93
d198	0.03	7.22	5.27
Mean	0.02	4.65	1.65
Standard Deviation	1	2	3

The results in the above tables display significant differences between the three algorithms. The LK algorithm results in an output that is over 2% closer to the optimum than the 3-opt. The 3-opt also runs almost 3 times as long as the LK algorithm. Since the LK algorithm performs than the 3-opt and the LK algorithm runs faster, the 3-opt is no longer a viable candidate.

The LK algorithm is on average over 5% more accurate than the 2-opt. However, the 2-opt is marginally the quickest iterative improvement candidate. The difference is substantial, even higher than expected when comparing the complexities of the 2-opt to the LK algorithm. It all comes down to whether the priority is speed or accuracy. As mentioned previously, the FBS trajectory computation time must be extremely fast, and the quality of the 2-opt is still good at the expected average of 5% and the experimented result of 6.28%. Note that the 2-opt will be nested inside a larger algorithm that will be used with the intention of optimizing the location of the terrestrial MBS cell edge. Therefore, a substantial speed advantage makes the 2-opt a more practical algorithm for the use case presented in this project. The TSP solving algorithm cannot afford longer running times since it will polynomially increase the complexity of the larger algorithm that will be presented in the following section. Also, the TSP solver will be run twice per experimental instance as the algorithm that will be presented will be dealing with two FBS's. Hence, speed must be secured at all reasonable costs. For these reasons, the 2-opt will be the chosen algorithm to improve on the NN solution to optimize the FBS

trajectory.

3.4 Cell Edge Location Optimization: Introducing the MTSP and Clustering

When appointing a FBS to serve users inside a terrestrial MBS cell, the 2-opt will be efficient in routing the FBS to optimize the trajectory. However, a new problem arises when two FBSs serving ground users are operating near the boundaries of two adjacent cells in a multi-cell environment. The situations where a flash-crowd is overcrowding a cell or a single ground node (GN) representing a cluster of users is on the boundary of a cell will be examined. Consider two FBSs that will be named FBS 1 and FBS 2 for simplicity where FBS 1 is appointed to serving GNs in cell 1 and FBS 2 is appointed to serving GNs in cell 2. If FBS 1 was directed to serve a large portion of GNs in cell 1, but then had to extend the route to the boundary of the cell to serve a single GN on the cell edge separating cell 1 and cell 2, the route cost of FBS 1 will be significantly increased for a single GN. It may be that FBS 2 is better suited to serve that GN due to the distribution of users in cell 2. However, consequent to the location of the cell edge, FBS 2 will not be appointed to serving that GN and FBS 1 will have to serve the GN. This will substantially increase the net cost of both FBSs. Had the cell edge been flexibly located so that the GN would have been inside cell 2, FBS 2 would have served the GN, decreasing the net cost of both FBSs. Similarly, if cell 1 is overcrowded with a flash-crowd that is distant from the cell edge, FBS 1 will need to be directed so that the flash-crowd can be served with maximal spectral efficiency. FBS 1 in this case will face the same problem previously mentioned. Serving users that are near the cell edge will compromise the trajectory cost and the FBS communications relative to the flash-crowd. A smartly located cell edge would optimize the problem by clustering the GNs near the cell edge to FBS 2, improving the data transmission links from FBS 1 to the flash-crowd in cell 1 and the total cost.

Due to such scenarios, the cell edge must be located according to the user distribution in the cells so that the FBSs can optimize the trajectories to minimize the total cost. To do this, the GNs must be clustered into two groups with each representing the mission area of a single FBS, based on the user locations in the multi-cell space. The TSP initally faced by a single FBS is now converted into a variation of a generalization of the TSP: the Multiple Traveling Salesman Problem (MTSP). It is no longer sufficient to find the optimal tour of a TSP to claim

trajectory optimization. The requirement now advances to having to solve two simultaneous TSPs so that both tours collectively cover the least distance possible.

3.4.1 Description

Clustering is the objective of grouping objects into multiple clusters based on the similarities between the objects. An efficient implementation of clustering will result in different clusters in which the objects have similarities that are different to those of objects in other clusters.[wiki clustering]

To solve the problem faced by the two FBSs, (3.5) will present an algorithm that will be devised to cluster the GNs using a local search technique. This will assist in splitting the two GN clusters according to the locations of the GNs in space. By using the local search techniques the clustering will focus on achieving the best possible multi-cell partitioning so that the MTSP problem faced by the FBSs is solved. In the standard MTSP, $m \geq 1$ salesmen must leave from one depot and return to it after having completed m tours subject to each tour passing through a city only once. Every city mast have been part of a single tour. The objective is that the resulting cost which is the addition of the distances covered by the tours is the lowest cost possible. Consequently, the tours must be optimally directed. The variation faced by the FBSs differs from the standard MTSP because in the case of the FBSs, each FBS must return to the MBS that the FBS was appointed to. Hence, rather than all tours having to end at a single depot, each tour can end in a different depot subject to every tour returning to the depot it originated from.

To assess the performance of the proposed algorithm, the results will be compared to those produced by similarly integrating the local search algorithm with the traditionally used K-means clustering algorithm. K-means clustering is effective in clustering object sets with using a technique that divides data sets based on centroids. [17] states that the K-means algorithm produces clustering results better than other algorithms.

3.5 Description of Simulation Schemes

To efficiently cluster all GNs as previously mentioned, some algorithms will be proposed and analysed. The simulations that intend to find the optimal cell edge location by implementing

the algorithms will then be explained.

3.5.1 Summary of Traditional Algorithms

As previously discussed, the 2-opt local search applied to a NN tour will be used in the algorithms that will be proposed. The 2-opt NN algorithm follows the instructions in the structure presented in the below pseudocode [20].

Algorithm 1 2-opt NN

Input: 1. (n points) 2. (xy coordinates of points) 3.(Starting NN tour

Output: 1. Tour between n points

```
Evaluate the distance matrix
Define the NN tour
for i = 1 : n − 2 do
    for j = i + 2 : n do
        evaluate total length d_1 of two edges
        evaluate total length d_2 of two edges when edges are swapped according to a 2-opt move
        d_2 < d_1 implement edge swap
    end
end
```

K-means Clustering will be used as the standard of evaluation of the proposed algorithms to verify the efficiencies and performances of the algorithms. K-means uses the spatial distribution of n points to partition the points in to k clusters. The algorithm works by randomizing locations of k centroids and calculating the distances between every point and every centroid, assigning points to the clusters centered at the centroids closest to the points [20]. The algorithm then replicates new centroids located at the centres of gravities of the points assigned to each cluster. The algorithm completes once the centroids are located at the mean of all points in the cluster assigned to the centroid, and no more changes can be made. K-means is known to have a complexity $O(n^2)$ The pseudocode of the algorithm is below [25].

Algorithm 2 K-means Clustering

Input: 1. (k - the number of clusters) 2. (D - a set of lift ratios)

Output: 1. Set of k clusters

Method:

Arbitrarily choose k centroids as inital cluster centers

Repeat:

1. assign each data point to the cluster with the closest centroid
2. update the centroid locations by finding the new mean values of the clusters

Until: no more updates possible

3.5.2 Analysis of Proposed Algorithms

Consider a two dimensional coordinate plane $P \subset \mathbb{R}^2$ in Euclidean space and two coordinates x, y where the point $(x, y) \in P$ where x and y are the largest values on the X-axis and the Y-axis respectively. The area size of P is $x \times y$ and there is a distribution of n points on the plane where each point $p \in \{p_1, p_2, ..., p_n\}$ and $\{\forall p \exists ! (x_p, y_p) | (0 \leq x_p \leq x) \wedge (0 \leq y_p \leq y)\}$. Both x and y in the following algorithms are equal to the Euclidean distance between two terrestrial MBSs.

Algorithm 1: FBS Linear Boundary-search

Algorithm 1 intends to find the optimal location of the linear cell edge that separates two cells by finding the net distance covered, using the 2-opt NN algorithm, by each FBS at multiple cell edge locations. The output will be the cell edge location at which the net distance is the minimum net distance. This will group the GNs within the cells into two clusters each of which will be serviced by the FBS appointed to the terrestrial MBS in the according cell. Algorithm 1 iterates through 7 X-coordinates representing a linear boundary $b_q \in \{b_1, b_2, ..., b_7\}$ where b_q is perpendicular to the X-axis at $x_{qi} | \frac{7x}{20} \leq x_{qi} \leq \frac{13x}{20}$. The algorithm does so by incrementally increasing the value of qi within the range $(\frac{7x}{20}, \frac{13x}{20})$ by $\frac{x}{20}$ at every iteration .

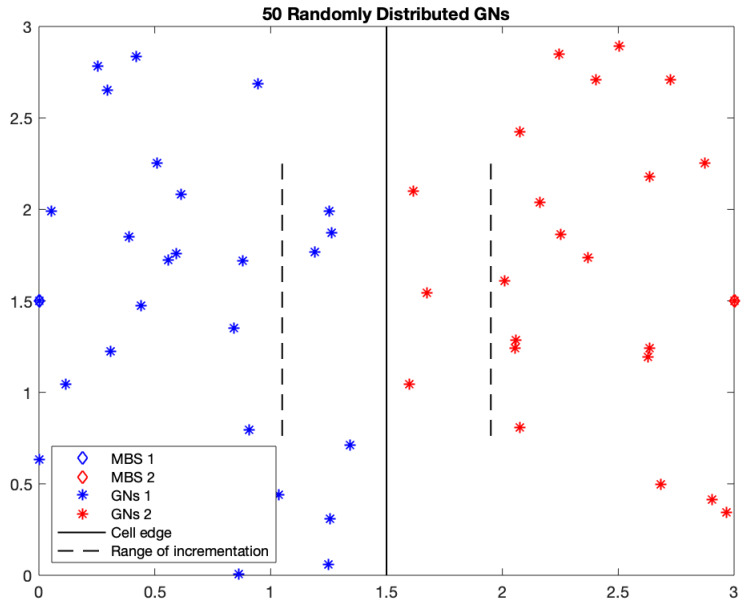

Figure 3.5: Boundary Incrementation Range

Figure 3.5 can be used to visualize how the algorithm will run. The net distance covered by the FBSs will be calculated at seven locations of the boundary. The first boundary is located at the dashed line on the left and the last at the dashed line on the right An explanation of the algorithm inputs and outputs is below followed by the pseudocode on the following page.

Input:

1. Is the value of the distance (km) between the two terrestrial MBSs.
2. Is the number of GNs to be served by the FBSs.
3. Is input as a 1×2 matrix [power(W) velocity(m/s)].

Output:

1. Is a 1×7 matrix containing the net distance (km) covered by the FBSs at every boundary location.
2. Is a 1×7 matrix containing the energy (J) consumed by FBS 1 at every boundary location.
3. Is a 1×7 matrix containing the energy (J) consumed by FBS 2 at every boundary location.
4. Is a 2×2 matrix containing the coordinates of the boundary location with the least net distance min(1) such that if the matrix was plotted with a line intersecting the two points the boundary would be the output.

5. Is the X-coordinate of the optimal boundary location.

Algorithm 3 FBS_Boundary_search_lin

Input: 1. (cell_diamater) 2. (GNs) 3. (UAV_parameters)

Output: 1. (dnet_lin) 2. (ec1_lin) 3. (ec2_lin) 4. (coordinates_optimal_edge_lin) 5. (x_optimal_edge_lin)

Add MBS coordinates to GNs matrix

for *B = 1:7* **do**

    Increment X-coordinate of boundary

    Assign points left of new boundary

    Implement 2-opt search to the assigned points

    Find energy consumed by FBS 1

    X = M(x_M>x_boundary(1),:);*%Build matrix of points right of new boundary*

    Assign points right of new boundary

    Implement 2-opt search to the assigned points

    Find energy consumed by FBS 2

end

Output optimal edge index

return Energy consumption of FBS 1,2 and Net distance and Optimal boundary coordinates

The 2-opt algorithm firstly implements the greedy NN algorithm, which has a computational complexity of $O(n^2)$, to find a starting tour. The 2-opt algorithm, which also has a complexity of $O(n^2)$, is then implemented to improve the initial tour. As observable in the 2-opt NN pseudocode, the 2-opt begins running after NN runs, and so the exact worst case running time is $(2 \times n^2)$. The constant 2 will be disregarded as $n \to \infty$ in the worst case. Therefore, the computational complexity of the 2-opt NN algorithm is $O(n^2)$. As observable in the pseudocode of Algorithm 1, the 2-opt runs simultaneously within the boundary iteration loop. The loop runs from B = 1:7, hence, the exact worst case running time is $(7 \times n^2)$. Considering a constant upper bound of 7 for B, the constant 7 will be disregarded as $n \to \infty$ in the worst case. Therefore, the underlying computational complexity of Algorithm 1 , where n is the size of GNs, is $O(n^2)$. The algorithm runs in polynomial time, making the implementation quick.

Algorithm 2: FBS Piece-wise Boundary-search

Algorithm 2 intends to further optimize the cell edge location by improving the output of Algorithm 1. The algorithm does so by inputting the optimal linear cell edge location and outputting an improved piecewise cell edge, further optimizing the GNs clustering. The same general procedure that Algorithm 1 follows is followed by Algorithm 2. The algorithm does so by initially using the output from Algorithm 1: the X-coordinate of the optimal linear cell edge location. Algorithm 2 then splits the linear boundary at two points p_2 and p_3 located at $(\frac{x}{2}, \frac{3y}{4})$ and $(\frac{x}{2}, \frac{1y}{4})$ respectively. Algorithm 2 then iterates through five different locations of p_2 and does so for $p3$ at every location of $p2$. Both $p2$ and $p3$ are incremented within the range $(\frac{7x}{20}, \frac{13x}{20})$ as observed in Figure 3.6 below. Algorithm 2 will then iterate through 25 different piecewise boundary locations by incrementally increasing each point by $\frac{x}{40}$ and output the boundary at which the net distance covered by the FBSs is the lowest.

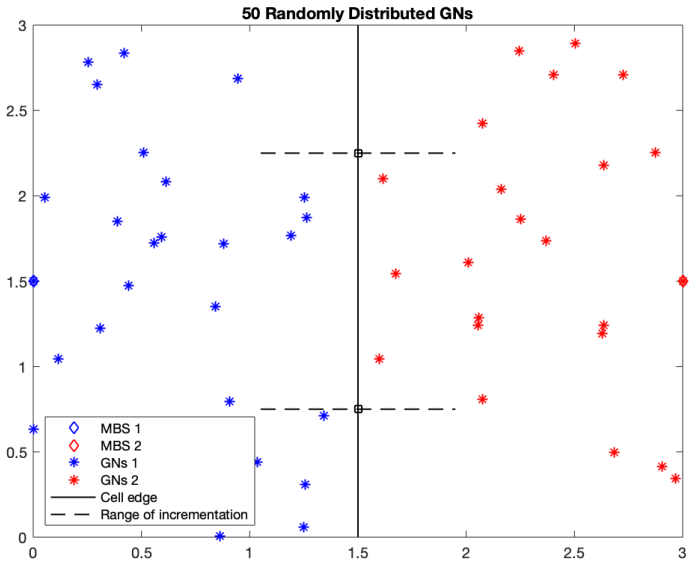

Figure 3.6: Point Incrementation Range

Input:

1. Is the value of the distance (km) between the two terrestrial MBSs.
2. Is the number of GNs to be served by the FBSs.
3. Is input as a 1×2 matrix [power(W) velocity(m/s)].
4. Is the X-coordinate of the optimal linear boundary location.

Output:

1. Is a 1×7 matrix containing the net distance (km) covered by the FBSs at every piecewise boundary location.
2. Is a 1×7 matrix containing the energy (J) consumed by FBS 1 at every piecewise boundary location.
3. Is a 1×7 matrix containing the energy (J) consumed by FBS 2 at every piecewise boundary location.
4. Is a 4×2 matrix containing the coordinates of points at the starting point, endpoint, and the points at which the piecewise boundary with the least net distance min(1) splits such that if the matrix was plotted with a line intersecting the four points the piecewise boundary would be the output.

Algorithm 4 FBS_Boundary_search_pw

Input: 1. (cell_diamater) 2. (GNs) 3. (UAV_parameters) 4. (x_optimal_edge_lin)

Output: 1. (dnet_pw) 2. (ec1_pw) 3. (ec2_pw) 4. (coordinates_optimal_edge_pw)

Add MBS coordinates to GNs matrix

for *p2 = 1:5* **do**

    Increment X-coordinates of p2

    for *p3=1.5* **do**

        Increment X-coordinates of p3

        Build a polygon bounded by piecewise boundary enclosing area left of piecewise boundary;

        Implement 2-opt on points inside the polygon

        Find energy consumed by FBS 1

        Implement 2-opt on points not inside the polygon

        Find energy consumed by FBS 2

        Group all net distances in matrix

    end

end

Find optimal edge index

return Energy consumption of FBS 1,2 and Net distance and Optimal boundary coordinates

As previously discussed, the computational complexity of the 2-opt NN algorithm is $O(n^2)$. As observable in the pseudocode of Algorithm 2, the tspsearch runs simultaneously within the p_3 iteration loop. The p_3 iteration loop runs simultaneously within the p_2 iteration loop. Both loops p_2 and p_3 run from p_2, p_3 = 1:5, hence, the exact worst case running time is $(25 \times n^2)$. Considering a constant upper bound of 25 for $p_2 \times p_3$, the constant 25 will be disregarded as $n \to \infty$ in the worst case . Therefore, the underlying computational complexity of Algorithm 2 , where n is the size of GNs, is $O(n^2)$. The algorithm runs in polynomial time, making the implementation quick.

Algorithm 3: FBS Boundary-search

Algorithm 3 combines both Algorithms 1,2 to efficiently find the optimal piecewise edge with only one input layer. The algorithm uses the input of Algorithm 1 and outputs the output of Algorithm 2. The pseudocode is as follows.

Algorithm 5 FBS_Boundary_search

Input: 1. (cell_diamater) 2. (GNs) 3. (UAV_parameters)

Output: 1. (dnet_pw) 2. (ec1_pw) 3. (ec2_pw) 4. (coordinates_optimal_edge_pw)

41 Implement Algorithm 1 to find optimal linear edge

42 Implement Algorithm 2 to find optimal piecewise edge

43 **return** Energy consumption of FBS 1,2 and Net distance and Optimal boundary coordinates

As observable in the pseudocode above, Algorithm 2 begins running after Algorithm 1 runs, and so the exact worst case running time is $(2n^2)$. The constant 2 will be disregarded as $n \to \infty$ in the worst case. Therefore, the underlying computational complexity of Algorithm 3 , where n is the size of GNs, is $O(n^2)$. The algorithm runs in polynomial time, making the implementation quick.

Analysis of Simulations

The simulations that will be discussed in the remainder of Chapter 3 collectively intend to simulate real-life scenarios by experimenting with different GN distributions. The process will obtain results that address the previously discussed problems that are encountered in multi-cell environments. However, approximating a realistic result is highly improbable with only a single distribution of users. This is due to the deterministic nature of the optimal boundary problem. A solution cannot be obtained through a random sample of GNs and be considered a viable solution for a different distribution of GNs. Therefore, the following simulations will all run using the Monte Carlo method. The method relies on repeated random sampling to obtain viable results. the Cambridge Dictionary of Statistics broadly defines the Monte Carlo method as the artificial generation of random processes to imitate the behavior of particular statistical models [12 bspr]. The method is highly effective in using randomness to solve deterministic problems [wiki montecarlo]. The number of GNs distributed will vary within every simulation to analyze a wider. range of results. The approaches and parameters used will be further explained in the following discussions.

Simulation 1: Linear Boundary

Simulation 1 implements Algorithm 1 to approximate the location of the optimal linear cell edge between two adjacent cells with an even distribution of GNs. The parameters involved with this simulation are the number of Monte Carlo runs and the number of node iterations. Simulation 1 will run for 1,000 Monte Carlo instances with each instance seeding a newly

randomized distribution of GNs. The size of the Monte Carlo simulation was settled with after analyzing results of differently bounded simulations. The conclusion can be summarized in the below figure.

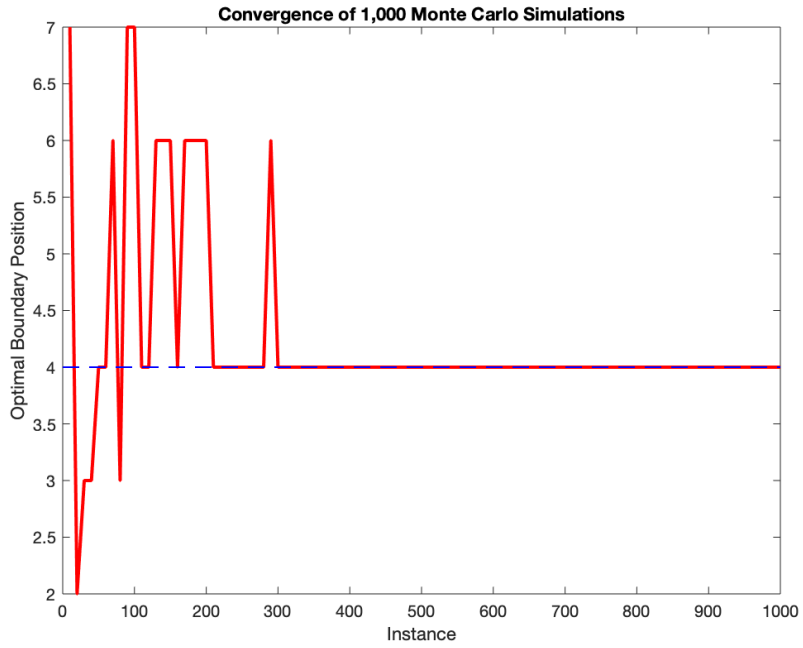

Figure 3.7: Pattern of 1,000 Monte Carlo simulations

The figure displays the convergence of all results from 1 to 1,000 runs in increments of 10. The Y-axis result of each instance on the X-axis is the obtained linear optimal boundary index after using Algorithm 1 for every instance with a different distribution of GNs. As observable, a pattern is formed after approximately 300 instances of the Monte Carlo simulation. The results are consistent afterwards, meaning that any size that is larger than 300 is reasonably sufficient for the simulation. For these reasons, a size of 1,000 runs was selected.

The size of the GNs was chosen to iterate from 20 to 80 in increments of 10. This is particularly because the results that will be presented in Chapter 4 displayed consistent patterns from 50 GNs onwards for some simulations, and displayed results for 20 nodes that could be used to draw conclusions on the relation of the results to the number of distributed GNs. The power and velocity of the UAV were obtained from paper [energy minimization vasilis]. The parameters of Simulation 1 are summarized in the following table.

Table 3.8: Simulation 1 Parameters

Parameter	Description	Value
MC	Monte Carlo runs	1000
GNs	Ground Nodes	20:10:80
P(W)	UAV Power	50
V(m/s)	UAV Velocity	60
A(km^2)	(Distance between MBSs)^2	3×3
U(a,b)	Uniform Distribution of GNs	

For every node size, the simulation will run 1,000 instances, hence, 7,000 in total. The pseudocode of the simulation is as follows.

Algorithm 6 Simulation 1

Input: 1. (nodes = 10) 2. (cell_diameter = 3) 3. (seed = 0) 4.(UAV_parameters = [50 60])

Output: 1. (Optimal linear boundary)

```
for nodes = 1:7 do
    Increment nodes by 10
    for i = 1:1000 do
        increment seed
        seed the rand generator
        generate random GNs
        Implement Algorithm 1 to find optimal linear edge
         Find energy consumed by FBS 1
        Find energy consumed by FBS 2
    end
end
return All results
```

As previously concluded, Algorithm 1 has complexity $O(n^2)$. The exact worst case running time for Simulation 1 is $(7 \times 1000 \times n^2)$. However, since algorithm complexity assesses the running time as a function of the primary input, which is the number of GNs in this simulation, the constants will be disregarded as $n \to \infty$ in the worst case. If nodes and Monte Carlo runs are variables, the complexity would be $O(n^4)$. However, they are constants in this simulation and the primary input is only the size of the GNs. Therefore, the underlying computational complexity of Simulation 1 , where n is the size of GNs, is $O(n^2)$. The simulation runs in polynomial time, making the implementation quick.

Simulation 2: Piecewise Boundary

Simulation 2 implements Algorithm 2 to approximate the location of the optimal piecewise cell edge between two adjacent cells with an even distribution of GNs. Similarly to Simulation1, Simulation 2 will have the same parameters as those of Simulation 1 in Table (3.8). The differences in simulations alongside the algorithm used is firstly the comparison to the K-means clustering algorithm. Simulation 2 will test the performance of Algorithm 2 by comparing the results of the algorithm to those obtained when implementing the 2-opt NN algorithm to the K-means clusters. Secondly, simulation 2 will output the energy consumed for each FBS to also assess the effect of clustering on the energy consumption per FBS. The pseudocode of Simulation 2 is as follows.

Algorithm 7 Simulation 2

Input: 1. (nodes = 10) 2. (cell_diameter = 3) 3. (seed = 0) 4.(UAV_parameters = [50 60] 5. x_optimal_edge_lin = 1.5km)

Output: 1. (Optimal linear boundary)

```
for nodes = 1:7 do
    Increment nodes by 10
    for i = 1:1000 do
        increment seed
        seed the rand generator
        generate random GNs
        Implement Algorithm 2 to find optimal piecewise edge
         Find energy consumed by FBS 1
        Find energy consumed by FBS 2
        Implement kmeans Find energy consumed by FBS 1
        Find energy consumed by FBS 2
    end
end
return All results
```

As previously discussed, the exact worst case running time for Simulation 2 would be $(7 \times 1000 \times 2 \times n^2)$ when also considering the K-means algorithm. However, the constants will be disregarded as $n \to \infty$ in the worst case, and the underlying computational complexity of Simulation 2 , where n is the size of GNs, is $O(n^2)$. The simulation also runs in polynomial

time.

Simulation 3: Linear Boundary with Flash-crowd

Simulation 3 implements Algorithm 1 to approximate the location of the optimal linear cell edge between two adjacent cells with a piecewise linear distribution of GNs. The simulation follows the exact same pseudocode as Simulation 1, with only the GNs matrix generation differing. The matrix is computed by increasing the probability of a number in the range $(0.65, 0.85)$ of being generated by the random generator to 45% as shown in the below pseudocode.

Algorithm 8 Simulation 3 Probability Distribution

```
for nodes = 1:7 do
    Increment nodes by 10
    for i = 1:1000 do
        increment seed
        seed the rand generator
        generate random GNs according to piecewise linear distribution
        Implement Algorithm 1 to find optimal linear edge
        Find energy consumed by FBS 1
        Find energy consumed by FBS 2
    end
end
```

The 45% of GNs in the region in $(0.65, 0.85)$ will represent a flashcrowd in a$200m \times 3000m$ strip in cell 1. The strip can simulate a street, such as Oxford Street in London, United Kingdom, where the density of the street compared to the density in the borough containing the street is similar to the above distribution. Simulation 3 is bounded by the same parameters assigned to Simulation 1 excluding the probability distribution. Simulation 3 also has a computational complexity of $O(n^2)$.

Simulation 4: Piecewise Boundary with Flash-crowd

Simulation 4 implements Algorithm 2 to approximate the location of the optimal piecewise cell edge between two adjacent cells with a piecewise linear distribution of GNs. The simulation follows the exact same pseudocode as Simulation 2, with only the GNs matrix generation differing. The probability distribution is the same distribution used in Simulation 3, referred to in Figure (3.9).

Chapter 4

Numerical Investigations

Results, performance analyses and evaluations will be presented in this Chapter to analyze the performance of the previously proposed algorithms, and to further understand the multi-cell environment and the problems associated with it through numerical investigation.

4.1 Results: Simulation 1

In Simulation 1 the objective was to find the optimal location of a linear cell edge separating two adjacent cells. This was done to optimize the FBS trajectory and minimize the associated costs. The results of Simulation 1 are as follows.

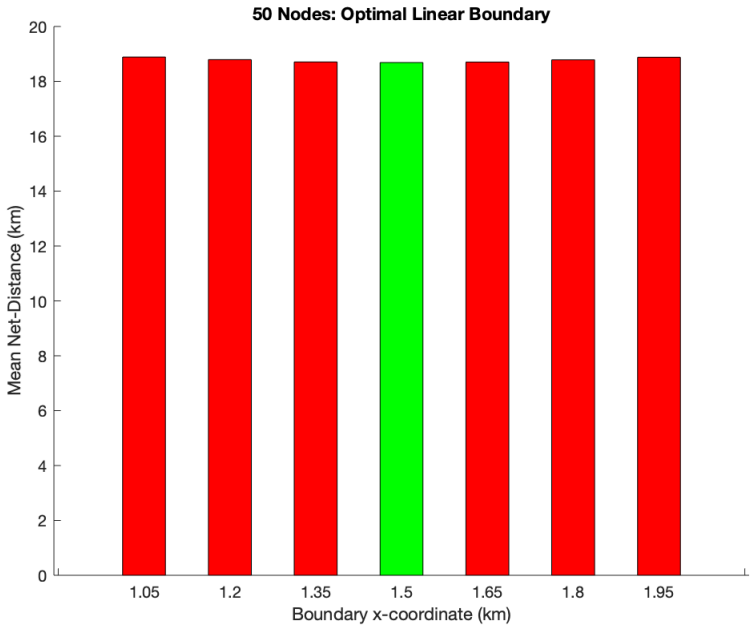

Figure 4.1: Optimal Linear Cell Edge Location after 1,000 Instances

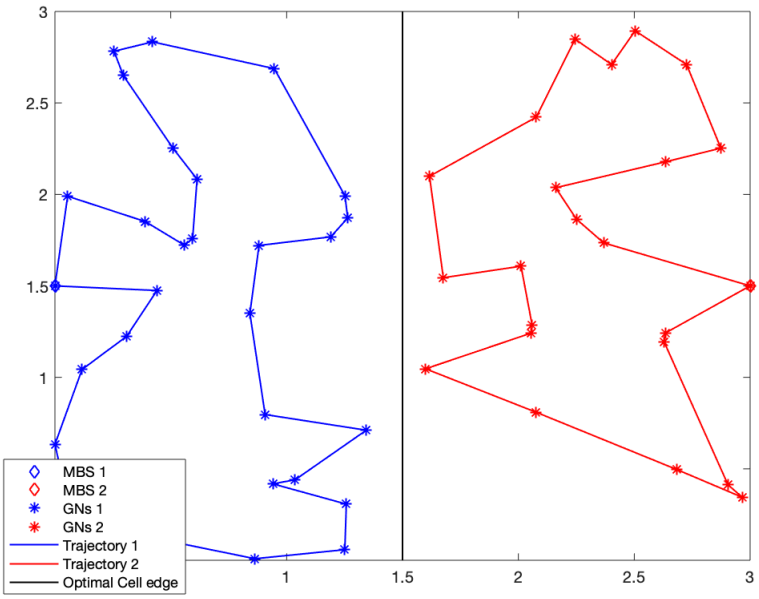

Figure 4.2: Optimal Linear Cell Edge for 50 GNs - Instance 1 Trajectory Representation

As expected and observed in the above figures, when uniformly distributing GNs the central boundary is the optimal location for the linear cell edge as the lowest mean net distance is at the boundary with an X-coordinate of 1.5km in the 3km axis.. However, Figure (4.2) displays a pattern of decreasing values from left to center. Within boundaries 3 and 5 there is a boundary location at which the mean net distances begin increasing and the results display a continual increase until the final location. Note that the exact location that has the lowest possible mean net distance, precisely before the increase of values begins, is the optimal location for the boundary. To further investigate the area, Simulation 2 will present the results of the optimal piecewise boundary, an extension of boundary 4 (1.5km) in Figure (4.1), within boundaries 3 and 5. The results of all other GN sets are the same as the above result and follow the same pattern discussed, and will therefore be displayed in a table below with the standard deviations of results of every node set.

Key to Table (4.3):

1. Is the X-coordinate of the point (x,0) at which the optimal linear boundary intersects the X-axis.

2. Is the mean net distance of all 1,000 instances at the optimal boundary location.

3. Is the standard deviation of the data set containing the seven mean net distances at each

boundary obtained after all instances for all GN sizes. An example of a data set is the data set presented in Figure (4.1) for 50 GNs.

4. Is the value of the standard deviation of the data set containing the 1,000 net distances of the boundary that had the highest standard deviation in the instances.

Table 4.3: Simulation 1 Summary of Results

GNs	20	30	40	50	60	70	80
(1) x(km)	1.5	1.5	1.5	1.5	1.5	1.5	1.5
(2) μ_l(km)	13.33	15.41	17.1	18.69	20.05	21.35	22.58
(3) σ_1(km)	0.0303	0.0720	0.0926	0.0829	0.0806	0.0601	0.0610
(4) σ_2(km)	1.1420	1.0426	0.9382	0.8787	0.8368	0.8431	0.8352

The results in the above table summarize all findings from implementing Algorithm 1 in Simulation 1 to find the most optimal linear cell edge location. The central boundary at 1.5km was found to be the optimal location for the linear boundary. The largest of seven boundary standard deviations of 1,000 instances was recorded for 20 GNs. The GNs size and the reliability of the results are directly proportional as with the increase of GNs size, the table shows a decrease in the value of the worst standard deviation of 1,000 runs with the exception of a discrepancy in σ_2 of 70 GNs.

4.1.1 Results: Simulation 2

In Simulation 2 the objective was to improve on the results of Simulation 1 by focusing on the search space within the three most optimal linear boundary locations. This was done by converting the optimal linear boundary into a three-segment piecewise boundary and obtaining the results of multiple piecewise boundaries within the optimal region in the X-coordinate range (1.35,1.65)km. This will further optimize the FBS trajectory and minimize the associated costs. The results of Simulation 2 are as follows.

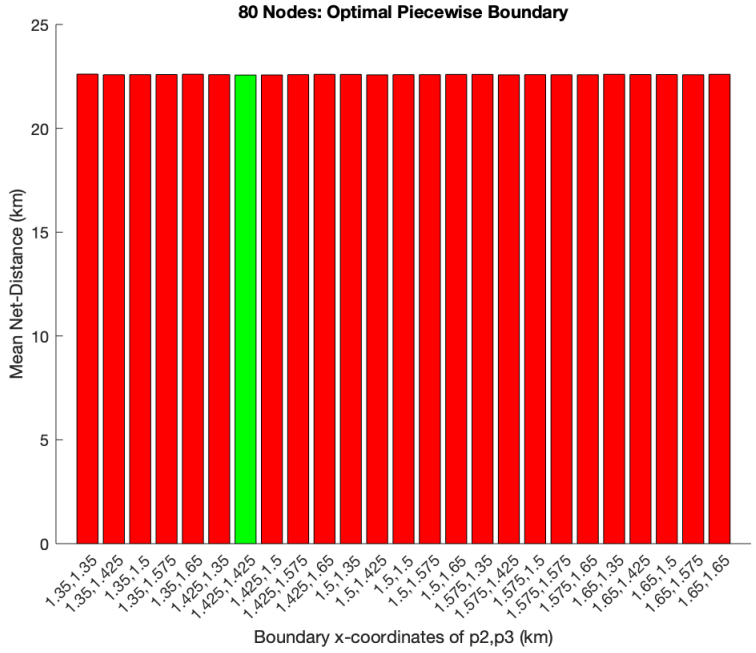

Figure 4.4: Optimal Piecewise Cell Edge Location after 1,000 Instances

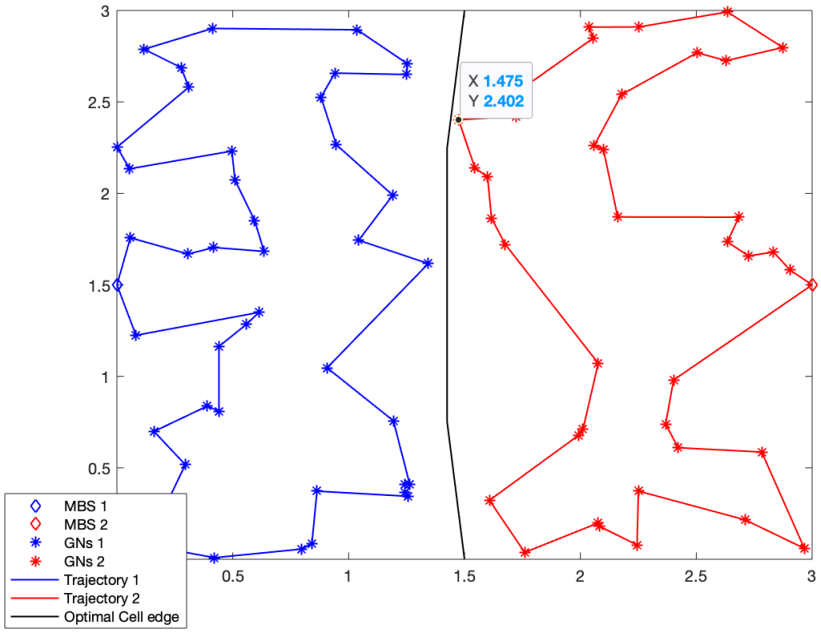

Figure 4.5: Optimal Piecewise Cell Edge for 80 GNs - Instance 1 Trajectory Representation

As observed in the above figures, when uniformly distributing GNs a piecewise cell edge is most optimal at coordinates [(1.5,3) (1.425,2.25) (1.425, 0.75) (1.5,0)]. The piecewise edge improved the net distance by approximately 20m. The most significant improvement was observable when experimenting with 80 nodes. This is expected as Algorithm 2 implements the

search in a very specific area to improve the local optimum obtained by Algorithm 1. The higher the density of GNs, the larger the effect of Algorithm 2. The algorithm effectively changed the cluster of the point with highlighted coordinates in Figure (4.4) and hence obtained an improved optimum. The algorithm is especially effective when dealing with large distances and GN sizes. As previously mentioned when discussing the cell edge placement problem, a single GN can make a difference to the trajectories of the FBSs. For these reasons, such algorithms can be implemented to reduce the costs by any means attainable. To test the clustering performance of Algorithm 2, a comparison of results to K-means clustering will be presented and discussed below.

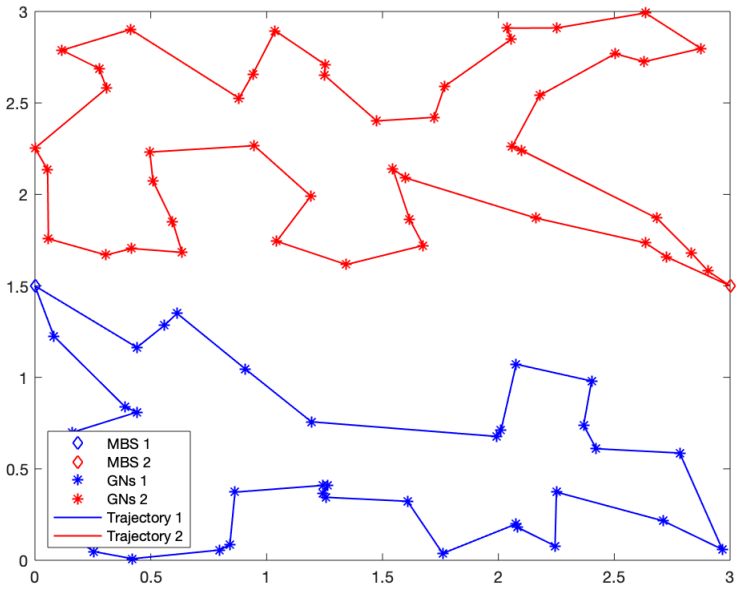

Figure 4.6: K-means clustering of 80 GNs at Instance 1

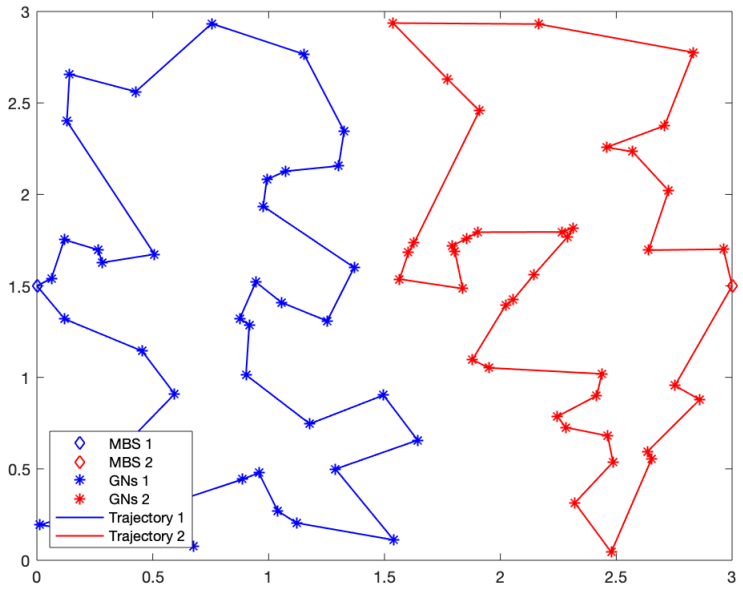

Figure 4.7: K-means clustering of 80 GNs at Instance 10

It is observable from the above figures that K-means clustering implements more dynamically than Algorithm 2. To assess the actual performances the below figures will compare the mean net distance of the optimal piecewise boundary obtained by Algorithm 2, to the mean net distance obtained by K-means clustering. The effect of the clustering methods on the energy consumption per FBS will also be assessed below.

Piecewise_Algorithm80.png

Figure 4.8: Comparison of Algorithm 2 and K-means Clustering - Mean Net Distance

The above figure displays a superiority in the performance of Algorithm 2 over K-means clustering. For all GN sets, the lowest mean net distance obtained by Algorithm 2 is lower than the mean net distance obtained by K-means clustering.

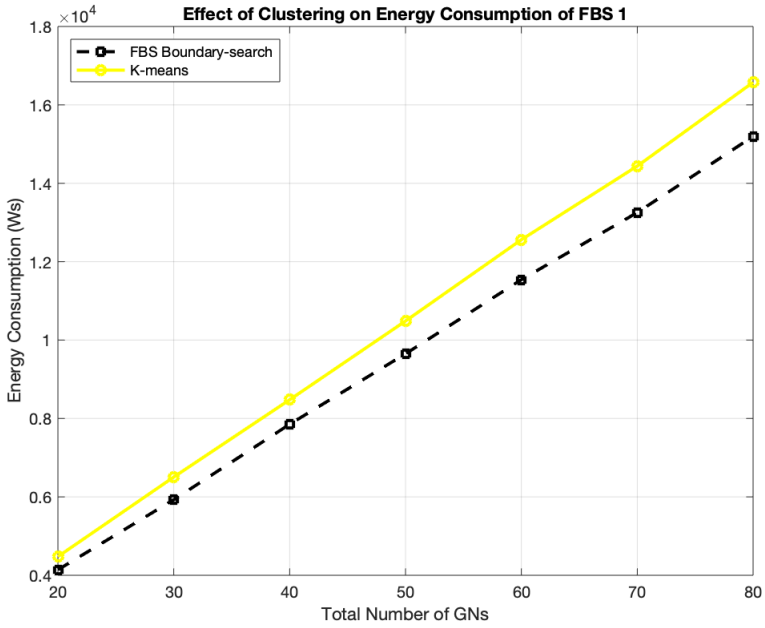

Figure 4.9: Effect of Clustering on Energy Consumption of FBS 1 after 1,000 Instances

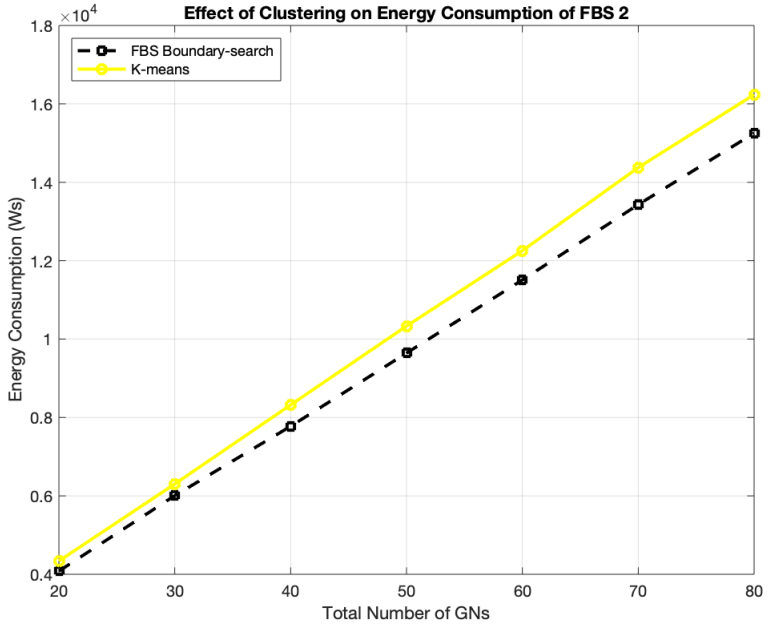

Figure 4.10: Effect of Clustering on Energy Consumption of FBS 2 after 1,000 Instances

Both FBSs consume less energy when GNs are clustered by Algorithm 2 for every size of GNs. The algorithm maintains a lower slope on both graphs than K-means by 9% for FBS 1 and 2% for FBS 2, highlighting the superiority in the algorithm's effectiveness with increasing numbers of GNs. The number of GNs increases the energy consumed as energy consumption of

a rotary-wing UAV is not limited to propulsion energy due to other energy consuming qualities of the UAV such as hovering. In the results, both propulsion energy and hovering energy are considered. For these reasons, the number of GNs increases energy consumption of an FBS not only because of the increased distance, but also because of the increased GNs to hover above. However, the FBSs both incur the effect of larger node sizes with lower significance when clustered by Algorithm 2 than when clustered by K-means. Algorithm 2 also maintains a stronger balance and consistency in the clustering as the rate of change in energy consumed per change in GN size is 7% higher for FBS 1 than for FBS 2. However, in the case of K-means clustering, the rate for FBS 1 surpasses the rate for FBS 2 by 14%. Algorithm 2 succeeds in minimizing effect of the GN size on the energy consumed, for both FBSs, when compared to K-means clustering. The algorithm does so by clustering more effectively and balancing the net FBS costs. The below table summarizes the results of Simulation 2.

Key to Table (4.11):

1. Is the X-coordinates of two points with coordinates (x_1,2.25) and (x_2,0.75)km at which the optimal piecewise boundary is incremented.
2. Is the mean net distance of all 1,000 instances at the optimal boundary location.
3. Is the decrease in distance from the output of Algorithm 1.
4. Is the standard deviation of the data set containing the 25 mean net distances at each boundary obtained after all instances for all GN sizes. An example of a data set is the data set presented in Figure (4.4) for 80 GNs.
5. Is the value of the standard deviation of the data set containing the 1,000 net distances of the boundary that had the highest standard deviation in the instances.
6. Is the mean net distance of all instances of the K-means clusters.

Table 4.11: Simulation 2 Summary of Results

GNs	20	30	40	50	60	70	80
Algorithm 2							
(1) (x_1, x_2)(km)	(1.425,1.575)	(1.5,1.5)	(1.5,1.5)	(1.5,1.5)	(1.5,1.575)	(1.5,1.425)	(1.425,1.425)
(2) μ_p(km)	13.32	15.41	17.1	18.69	20.04	21.35	22.56
(3) $\mu_l - \mu_p$(m)	10	0	0	0	10	0	20
(4) σ_1(km)	0.0072	0.0110	0.0129	0.0135	0.0106	0.0096	0.0105
(5) σ_2(km)	1.1071	0.9651	0.9091	0.8759	0.8267	0.8531	0.8550
K-means Clustering							
(6) μ_k(km)	13.54	15.68	17.33	18.87	20.21	21.45	22.64
(7) σ_k(km)	1.2972	1.0905	0.9728	0.9225	0.8441	0.8186	0.8080

4.1.2 Results: Simulation 3

In Simulation 3 the objective was to find the optimal location of a linear cell edge separating two adjacent cells. However, in this simulation the GNs were distributed according to a Piecewise linear distribution to model a flash-crowd. As discussed in Chapter 3, the flashcrowd will be represented by the approximate 45% of GNs inside a $200m \times 3000m$ area. This was done to explore an overcrowded cell in a multi-cell environment. The objective will again be to optimize the FBS trajectory and minimize the associated costs, and to approach the problem by taking a user based clustering approach rather than considering the traditional distance based approach to separate the cells. The results of Simulation 3 are as follows.

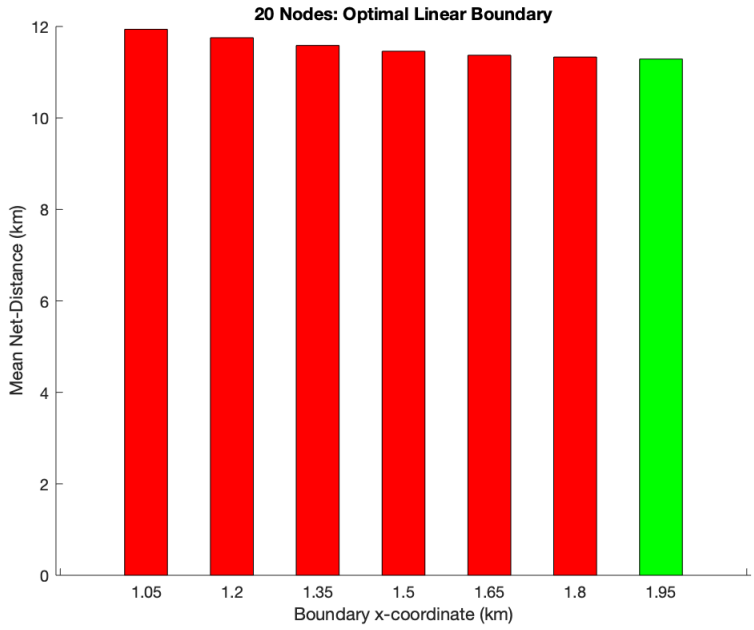

Figure 4.12: Optimal Linear Cell Edge Location after 1,000 Instances

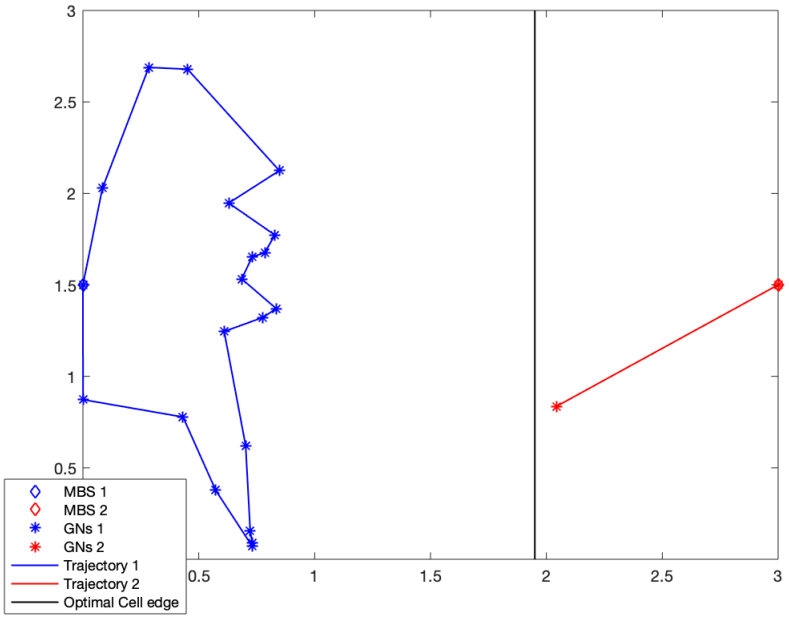

Figure 4.13: Optimal Linear Cell Edge Location at Instance 1 for 20 GNs with a Piecewise Linear Distribution - Instance 1 Trajectory Representation

The above results show the optimal region to be within (1.8,2.1)km on the X-axis as the optimal linear edge is located at 1.95km. To further investigate where exactly the optimal boundary placement is, the results from Simulation 4 will be used. However, note that the

above results suggests that it may be more optimal to deploy a single FBS when handling a flashcrowd that has attracted most of the users in the adjacent cells. Figure (4.13) can be used to model an instance where almost all users of a low density multi-cell environment have crowded in one specified area. This special case will be further discussed in 4.1.3 of this Chapter.

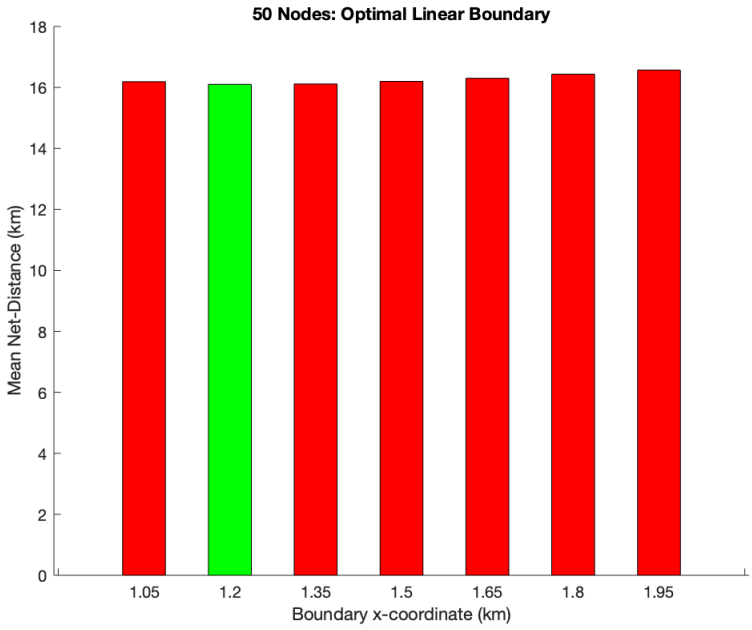

Figure 4.14: Optimal Linear Cell Edge Location

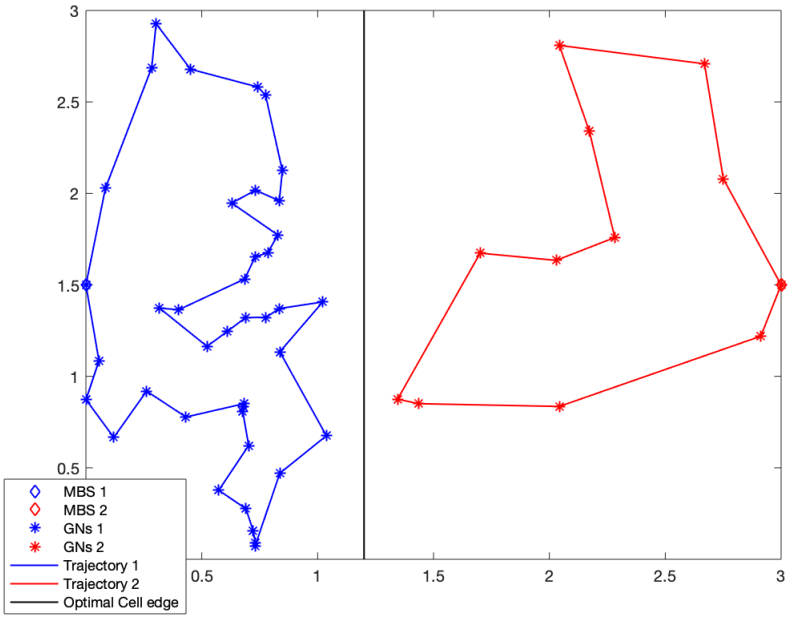

Figure 4.15: Optimal Trajectories of FBSs at Instance 1 for 50 Nodes

As observable in the above results, an increased number of GNs requires a focus of FBS

1's trajectory on the flash-crowd. When a cell is overcrowded with users, it is more optimal to focus the first FBS of the cell on the flash-crowd and use the second FBS to serve users far from the flashcrowd. To do so, the cell edge must be relocated according to the above figures. The below table summarizes the results of Simulation 3.

Key to Table (4.12):

1. Is the X-coordinate of the point (x,0) at which the optimal linear boundary intersects the X-axis.

2. Is the mean net distance of all 1,000 instances at the optimal boundary location.

3. Is the standard deviation of the data set containing the seven mean net distances at each boundary obtained after all instances for all GN sizes. An example of a data set is the data set presented in Figure (4.1) for 50 GNs.

4. Is the value of the standard deviation of the data set containing the 1,000 net distances of the boundary that had the highest standard deviation in the instances.

Table 4.16: Simulation 1 Summary of Results

GNs	20	30	40	50	60	70	80
(1) x(km)	1.95	1.5	1.35	1.2	1.2	1.2	1.2
(2) μ_l(km)	11.29	13.34	14.84	16.1	17.2	18.26	19.28
(3) σ_1(km)	0.2413	0.0947	0.1168	0.1748	0.2370	0.2875	0.3072
(4) σ_2(km)	0.1223	0.1397	0.1236	0.1206	0.0923	0.0515	0.0374

It can be concluded from the above figure that as the GN size increases, the optimal boundary location shifts more towards the flash-crowd location. This plans the FBS 1 trajectory to support the overcrowding in cell 1. All results for (2) display a significantly lower net distance than the results obtained when distributing GNs according to a uniform distribution in Simulation 1. This is because the overcrowding in Simulation 3 causes users to be closely clustered, minimizing the distances between GNs. The only problem encountered was when experimenting with 20 nodes, as if Figure (4.12) is assessed from left to right, the decreasing pattern reaches the upper bound of Algorithm 2's X-axis boundary increment range. As previously mentioned, this will be further analyzed in the 4.1.3. As observed when assessing all optimal boundary locations for all GN sizes , particularly the larger sizes, Algorithm 2 was highly effective in drastically relocating the cell edge location to support FBS 1's service of the flash-crowd. The

algorithm did so by limiting the trajectory of FBS 1 to the flash-crowd region on the expense of the trajectory of FBS 2. This was efficient in minimizing the net cost of deploying both FBSs.

4.1.3 Results: Simulation 4

In Simulation 4 the objective was to improve on the results of Simulation 3 by focusing on the search space within the three most optimal linear boundary locations. This was done by converting the optimal linear boundary into a three-segment piecewise boundary and obtaining the results of multiple piecewise boundaries within the optimal regions in the varying X-coordinate ranges for each set of GNs. This will further optimize the FBS trajectory and minimize the associated costs, assisting the overcrowded cell. The results of Simulation 4 are as follows.

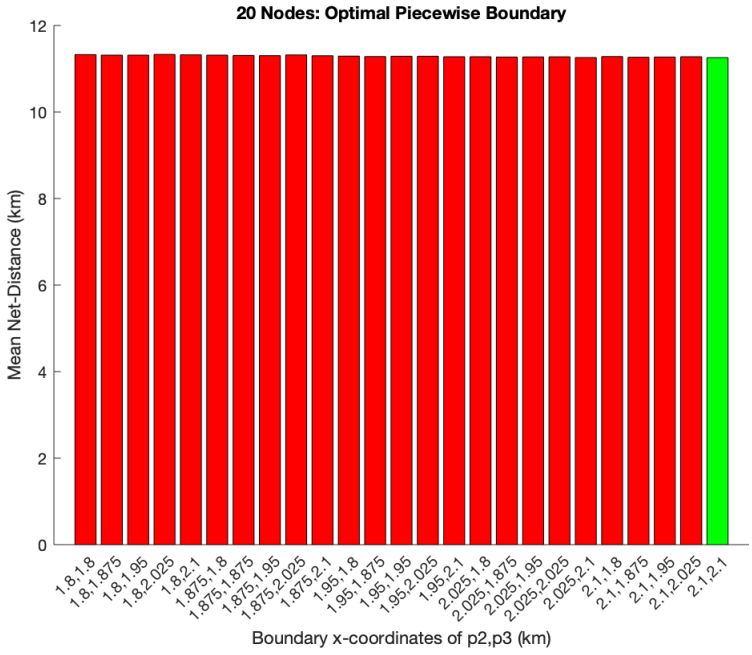

Figure 4.17: Optimal Piecewise Cell Edge Location after 1,000 Instances

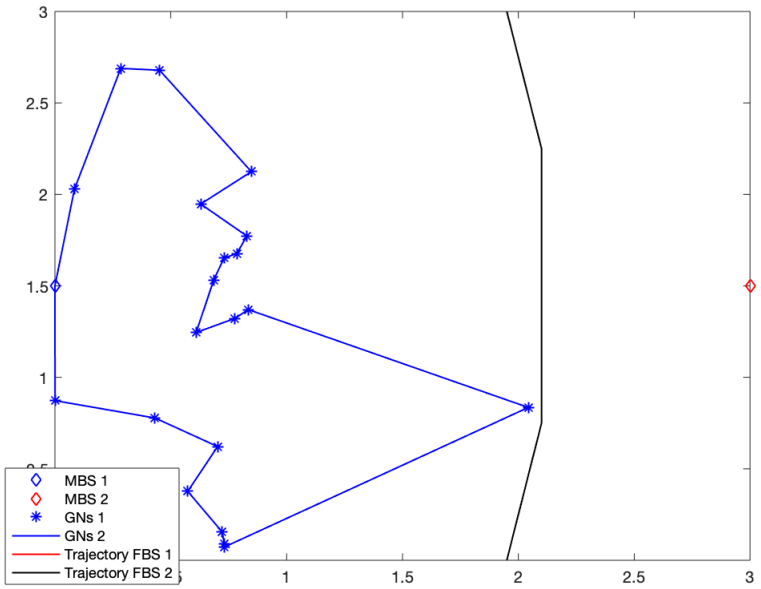

Figure 4.18: Optimal Piecewise Cell Edge for 20 GNs - Instance 1 Trajectory Representation

Figure (4.17) shows that when considering net distance, it is more optimal to deploy a single FBS to serve all users in the multi-cell space rather than deploying two FBSs. This is because as previously mentioned, the modelled event caused most of the users to crowd at the strip, emptying most of the remaining cell space. This makes the second FBS redundant as there are hardly any users to serve in cell 2 due to the overcrowding in cell 1. It can be concluded for multi-cell environments with less than 20 GNs that it is more optimal to deploy a single FBS when a flashcrowd containing a large portion of the users is gathered in a narrow region of a cell. However, for more active cells, a flashcrowd will not dominate the users across both cell spaces. Such examples will be further discussed below. This case could model a scenario where assisting of FBSs can also improve energy efficiency of the networks by reducing unnecessary consumption of energy. This can happen when a MBS is only serving few users. The FBS can then take over from the MBS by serving those users itself, allowing a fraction of the MBS to become inactive, saving energy and minimizing terrestrial interference in the network.

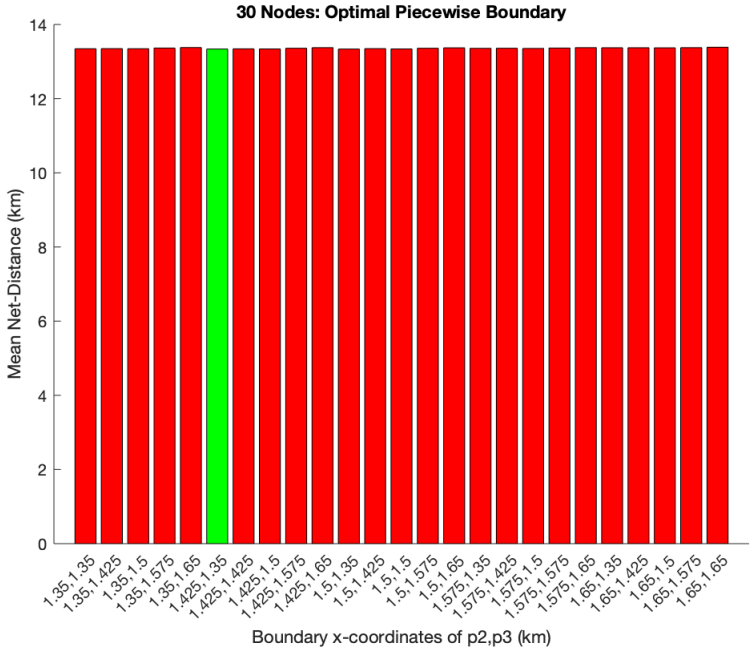

Figure 4.19: Optimal Piecewise Cell Edge Location after 1,000 Instances

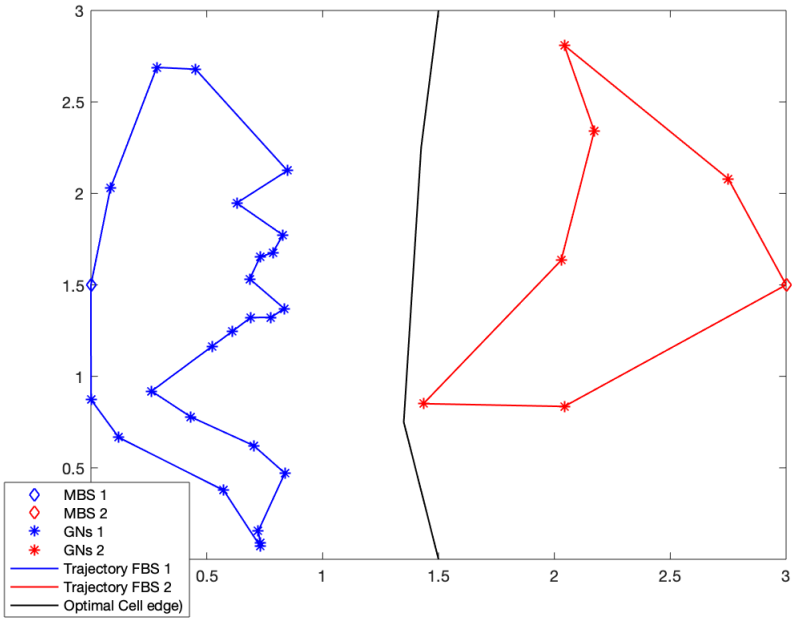

Figure 4.20: Optimal Piecewise Cell Edge for 30 GNs - Instance 1 Trajectory Representation

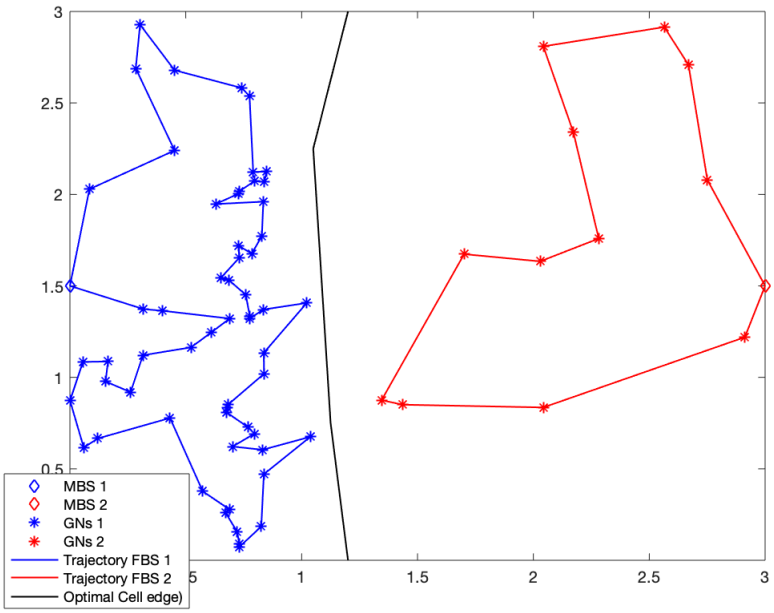

Figure 4.21: Optimal Piecewise Cell Edge for 70 GNs - Instance 1 Trajectory Representation

For 30 GNs, the optimal boundary location shifted from maintaining the central locations in both Simulations 1 and 3 to a location closer to the flash-crowd. The results displays that from 30 GNs onwards, all optimal boundaries move closer towards the flash-crowd rather than maintaining the traditionally positioned cell edge location. Figure (4.21) depicts a higher GN density scenario with the optimal edge at 70 GNs.

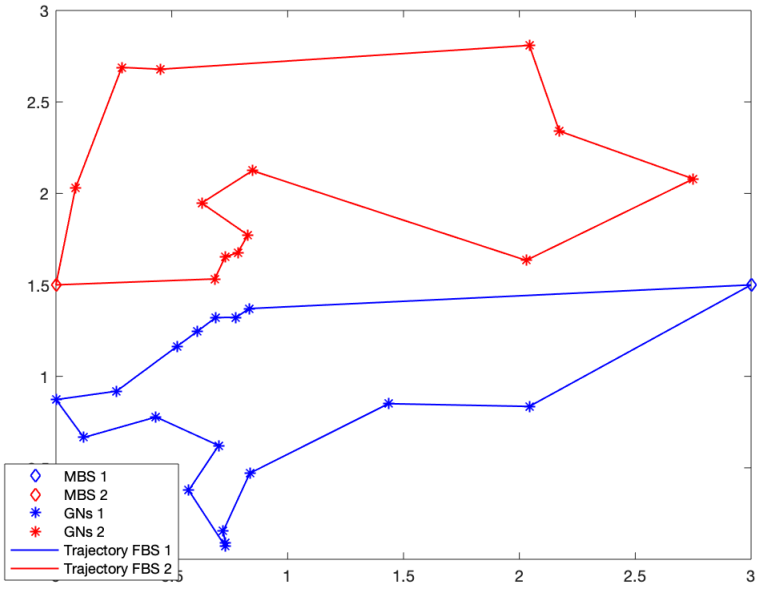

Figure 4.21: K-means Clustering of 30 GNs at Instance 1

K-means clustering considers a different approach as observed in the above figure. Rather than grouping the flash-crowd in a single cluster as Algorithm 2 does, K-means divides the flash-crowd into two clusters so that both FBSs are deployed to handle the overcrowding in cell 1. However, as the below results will confirm, Algorithm 2 was superior to K-means clustering when minimizing net distance covered by the FBSs.

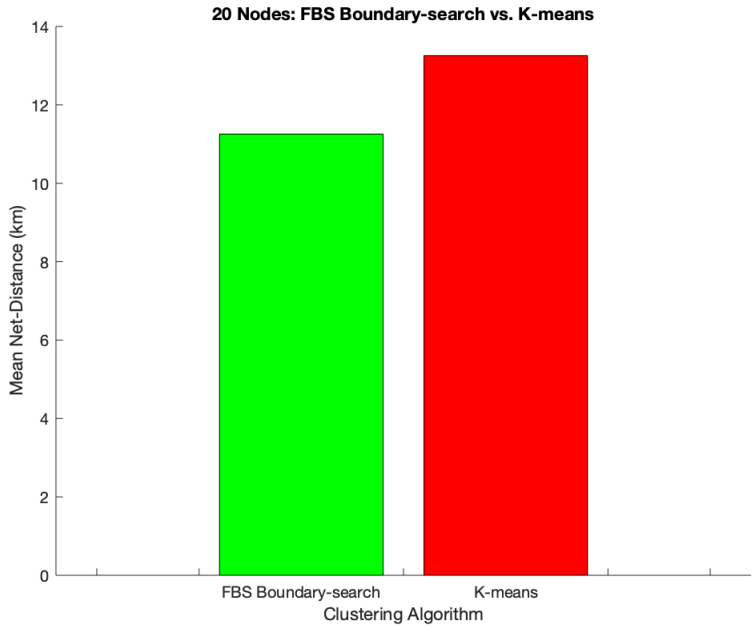

Figure 4.22: Comparison of Algorithm 2 and K-means Clustering - Mean Net Distance

The above result shows a significant difference in the performance between both algorithms. For all GN sizes, Algorithm 2 output a lower net distance than K-means clustering.

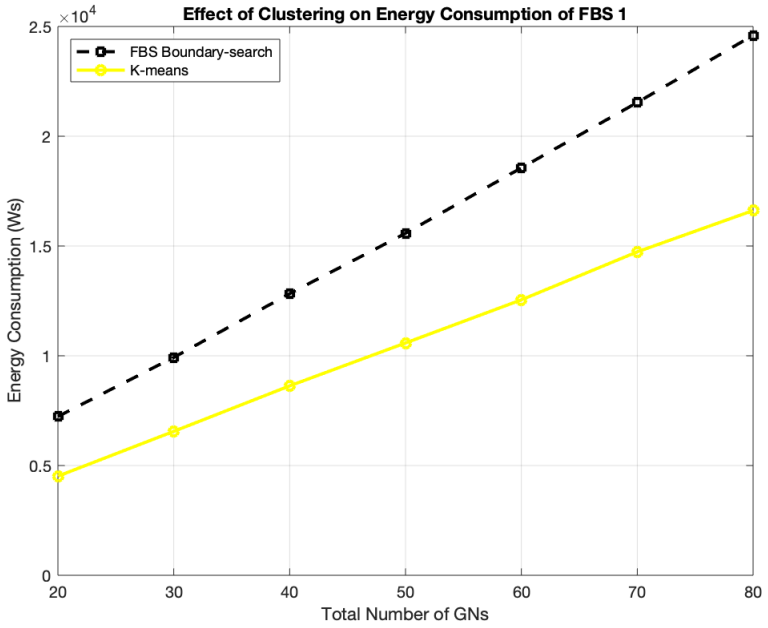

Figure 4.23: Effect of Clustering on Energy Consumption of FBS 1 after 1,000 Instances

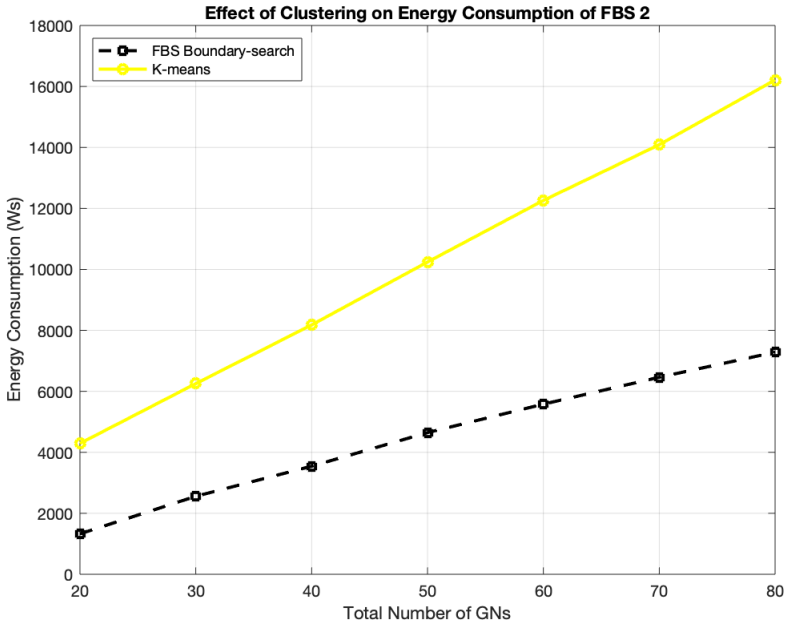

Figure 4.24: Effect of Clustering on Energy Consumption of FBS 2 after 1,000 Instances

The above results display dynamic advantages and disadvantages of the implementation of each algorithm when dealing with the flash-crowd scenario. When assessing the energy consumption of FBS 1, Algorithm 2 produces results that are significantly more detrimental to the FBS's energy consumption than K-means does. The algorithm also outputs a larger

increase in energy consumption for every increase in GN size than K-means. This is due to the slope of Algorithm 2 surpassing that of K-means by approximately 62%. However, K-means only managed to achieve better FBS 1 results by assigning flash-crowd segments in cell 1 to FBS 2. As observed in Figure (4.24), Algorithm 2 maintains a steady slope lower than the steeper slope of K-means by approximately 158%, due to the output of deploying FBS 2 to assist users far from the flah-crowd. Also, the maximum energy consumed by FBS 1 is 17.3 kJ higher than by FBS 2 for Algorithm 2. On the other hand, the difference for K-means is 410 J higher for FBS 1. However, Algorithm 2 maintains a lower net energy consumption of both FBSs. In conclusion, K-means succeeds in minimizing any overcrowded area by dividing the servicing mission amongst two FBSs. Algorithm 2 appoints FBS 1 to focus efforts on the flash-crowd region, and appoints FBS 2 to handle any GNs significantly outside the high density area. However, as displayed in Figure (4.22), the trade-off between the energy consumptions of each FBS was solved more optimally by Algorithm 2 as a lower distance was maintained at all GN sizes. The below table summarizes all Simulation 4 results.

Key to Table (4.25):

1. Is the X-coordinates of two points with coordinates (x_1,2.25) and (x_2,0.75)km at which the optimal piecewise boundary is incremented.

2. Is the mean net distance of all 1,000 instances at the optimal boundary location.

3. Is the decrease in distance from the output of Algorithm 1.

4. Is the standard deviation of the data set containing the 25 mean net distances at each boundary obtained after all instances for all GN sizes. An example of a data set is the data set presented in Figure (4.19) for 30 GNs.

5. Is the value of the standard deviation of the data set containing the 1,000 net distances of the boundary that had the highest standard deviation in the instances.

6. Is the mean net distance of all instances of the K-means clusters.

Table 4.25: Simulation 4 Summary of Results

GNs	20	30	40	50	60	70	80
Algorithm 2							
(1) (x_1, x_2)(km)	(2.1,2.1)	(1.425,1.35)	(1.2,1.275)	(1.125,1.125)	(1.05,1.125)	(1.05,1.125)	(1.125,1.0
(2) μ_p(km)	11.26	13.34	14.81	16.09	17.19	18.21	19.23
(3) $\mu_l - \mu_p$(m)	30	0	30	10	10	50	50
(4) σ_1(km)	0.0222	0.0148	0.0258	0.0218	0.0291	0.0529	0.0522
(5) σ_2(km)	1.6593	1.3884	1.1829	1.1579	1.0798	1.1011	1.0966
K-means Clustering							
(6) μ_k(km)	13.26	15.14	16.56	17.88	18.89	19.93	20.86
(7) σ_k(km)	1.5594	1.3610	1.1455	1.0719	0.9776	0.9601	0.9053 hei

The above results display a significant pattern of place the boundary near the flashcrowd for larger GN sizes. From 30 GNs and above, Algorithm 2 clusters the GNs such that FBS 1 is only required to handle cell 1's overcrowding. (2) is on average higher than (2) of Simulation 2, meaning that Algorithm 2 is particularly effective when applied to non-uniform GN distributions. K-means failed to achieve a lower net distance for all sets of GNs, with results also varying the a great extent per iteration than Algorithm 2's results.

4.1.4 Performance Analysis and Evaluation

Algorithm 1 displayed a performance capable of producing insight to relocate the cell edge according to the user distributions in the cells. The algorithm performed best when implemented in Simulation 3, as the cell edge was relocated significantly further from the traditional central location. Algorithm 2 was also efficient in further improving the results of Algorithm 1, however, the improvements were minor in these cases. The algorithm may be particularly effective when applied to larger distances, such as those covered by HAP UAVs, to larger sets of GNs. Simulation 4 showed that for 20 GNs and below, it may be more efficient to deploy a single FBS if an overcrowding contains around 45% of cell users in the crowd region. This is due to the multi-cell space having large distances with very low user densities due to the overwhelming event in the flash-crowd region. However, as discussed previously, this is a very special case that is applied to only certain scenarios. More generic results are obtained when experimenting on 30 GNs or more, because as expected, the algorithm plans the trajectory of FBS 1 so that

the FBS can serve only the flash-crowd. This will provide high QoS links for the flash-crowd. However, the algorithms are not as dynamic as K-means due to the search spaces. the figures highlighting the overlaid trajectories display significant varieties in clustering approaches when K-means is applied. As discussed and proven by every result obtained, both Algorithms still performed better than K-means clustering, particularly for the flash-crowd experiments. All results obtained displayed reliability through the standard deviation values of iterations per GN size, highlighting the consistency of 1,000 Monte Carlo runs. The following Chapter will discuss the potential improvements that can be made to the proposed algorithms.

Chapter 5

Conclusion and Future Work

The project addressed the potential of deploying UAVs as FBSs to assist 5G Networks. Trajectory optimization of FBSs was a key objective of this project, and hence intensive research on the TSP was discussed to better understand the problem and the approaches to solving it. After find an optimal route for a single FBS, a second problem arose: the optimal cell edge placement. This problem was dependent on the trajectories of both FBSs as the problem was primarily to group all GNs in the multi-cell space into two clusters so that each could be serviced by a FBS. The primary concern was when a flashcrowd overwhelms one cell, and this was the key motive of trying to find a more optimal location for the cell edge so that the GNs can be clustered based on the dispersion of users rather than the geographical locations of the terrestrial MBSs. An algorithm that finds an optimal linear cell edge location based on the user dispersion was proposed. A second algorithm that improves the results of the first was also proposed, and this algorithm successfully and consistently outperformed K-means clustering when tasked with clustering GNs the multi-cell space. Future works on the aforementioned topics could include the improvement of the algorithms so that more dynamic clusters can be found. This would be possible by rotating the cell edge at an angle at every iteration. This would then diversify the applications of the algorithms in future applications. The algorithms could also be used to cluster GNs within the same cell to determine the trajectories of multiple FBSs deployed at the same MBS. A further improvement that can be made to the experiments would be the application of K-means to find optimal hover locations, rather than hovering over all GNs. This would be significant in a flashcrowd event as the FBS would be able to serve denser crowd more easily from the same location. The limitations of the above works are firstly the algorithms, which did produce significant results against K-means, however, K-means is

more dynamic and can be applied to a multitude of applications. The current algorithms are very focused on cell spaces and not generic clustering. Some variations can be made to develop the algorithms so that they could be more applicable.

References

[1] Adnan Aijaz, Mischa Dohler, A Hamid Aghvami, Vasilis Friderikos, and Magnus Frodigh. Realizing the tactile internet: Haptic communications over next generation 5g cellular networks. *IEEE Wireless Communications*, 24(2):82–89, 2016.

[2] Mohammed A Alhanjouri and Belal Alfarra. Ant colony versus genetic algorithm based on travelling salesman problem. *Ant colony versus genetic algorithm based on travelling salesman problem*, 2(3), 2013.

[3] Daniel-Ioan Curiac. Towards wireless sensor, actuator and robot networks: Conceptual framework, challenges and perspectives. *Journal of Network and Computer Applications*, 63:14–23, 2016.

[4] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[5] Harpreet S Dhillon, Howard Huang, and Harish Viswanathan. Wide-area wireless communication challenges for the internet of things. *IEEE Communications Magazine*, 55(2):168–174, 2017.

[6] Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst case and probabilistic analysis of the 2-opt algorithm for the tsp. In *SODA*, pages 1295–1304. Citeseer, 2007.

[7] Michael Held and Richard M Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970.

[8] Ekram Hossain, Mehdi Rasti, Hina Tabassum, and Amr Abdelnasser. Evolution toward 5g multi-tier cellular wireless networks: An interference management perspective. *IEEE Wireless Communications*, 21(3):118–127, 2014.

[9] David S Johnson and Lyle A McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1(1):215–310, 1997.

[10] David S Johnson and Lyle A McGeoch. Experimental analysis of heuristics for the stsp. In *The traveling salesman problem and its variations*, pages 369–443. Springer, 2007.

[11] Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. The traveling salesman problem. *Handbooks in operations research and management science*, 7:225–330, 1995.

[12] Zoltán Ádám Mann. The top eight misconceptions about np-hardness. *Computer*, 50(5):72–79, 2017.

[13] Rajesh Matai, Surya Prakash Singh, and Murari Lal Mittal. Traveling salesman problem: an overview of applications, formulations, and solution approaches. *Traveling salesman problem, theory and applications*, 1, 2010.

[14] Arvind Merwaday and Ismail Guvenc. Uav assisted heterogeneous networks for public safety communications. In *2015 IEEE wireless communications and networking conference workshops (WCNCW)*, pages 329–334. IEEE, 2015.

[15] Mohammad Mozaffari. *Wireless Communications and Networking with Unmanned Aerial Vehicles: Fundamentals, Deployment, and Optimization.* PhD thesis, Virginia Tech, 2018.

[16] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, and Mérouane Debbah. Unmanned aerial vehicle with underlaid device-to-device communications: Performance and tradeoffs. *IEEE Transactions on Wireless Communications*, 15(6):3949–3963, 2016.

[17] Yirga Yayeh Munaye, Hsin-Piao Lin, Abebe Belay Adege, and Getaneh Berie Tarekegn. Uav positioning for throughput maximization using deep learning approaches. *Sensors*, 19(12):2775, 2019.

[18] Christian Nilsson. Heuristics for the traveling salesman problem. 01 2003.

[19] Christos H Papadimitriou and Kenneth Steiglitz. On the complexity of local search for the traveling salesman problem. *SIAM Journal on Computing*, 6(1):76–83, 1977.

[20] Anoop Sathyan, Nathan Boone, and Kelly Cohen. Comparison of approximate approaches to solving the travelling salesman problem and its application to uav swarming. *International Journal of Unmanned Systems Engineering.*, 3(1):1, 2015.

[21] Silvia Sekander, Hina Tabassum, and Ekram Hossain. Multi-tier drone architecture for 5g/b5g cellular networks: Challenges, trends, and prospects. *IEEE Communications Magazine*, 56(3):96–103, 2018.

[22] Vishal Sharma, Kathiravan Srinivasan, Han-Chieh Chao, Kai-Lung Hua, and Wen-Huang Cheng. Intelligent deployment of uavs in 5g heterogeneous communication environment for improved coverage. *Journal of Network and Computer Applications*, 85:94–105, 2017.

[23] Qingqing Wu, Yong Zeng, and Rui Zhang. Joint trajectory and communication design for multi-uav enabled wireless networks. *IEEE Transactions on Wireless Communications*, 17(3):2109–2121, 2018.

[24] Yong Zeng, Jie Xu, and Rui Zhang. Energy minimization for wireless communication with rotary-wing uav. *IEEE Transactions on Wireless Communications*, 18(4):2329–2345, 2019.

[25] Pei-Yuan Zhou and Keith CC Chan. A model-based multivariate time series clustering algorithm. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 805–817. Springer, 2014.