POLITECNICO DI TORINO

MASTER's Degree in Communications and Computer Networks Engineering

MASTER's Degree Thesis

Routing Strategies for LEO and VLEO Mega-Constellations: Performance Evaluation and Enhancements

Supervisors

Prof. ROBERTO GARELLO

Dr. ALESSANDRO COMPAGNONI

Candidate

CAMILLA OTTAVIANI

OCTOBER 2024

Summary

In today's world it is possible to deploy thousands of satellites at lower orbits, due to the rapid technological advancements in the aerospace engineering field and the drop in the manufacturing costs of satellites. Non Geo-stationary Satellite Orbit (NGSO) mega-constellations, consisting of Low Earth Orbit (LEO) and Very Low Earth Orbit (VLEO) satellites at altitudes between 300 km and 2000 km, present new challenges when addressing routing. Mainly, they arise from the configuration of the constellation under test and the selection of paths, which require innovative routing solutions to ensure efficient data transmission. In the following research work, the performance of different deterministic routing algorithms has been studied for both LEO and VLEO Walker Delta constellations as well as the impact of distinct altitude, inclination and topology. Additionally, a novel heuristic has been introduced, showing its potential to significantly reduce overall latency by optimizing routing patterns in the Arctic and Antarctic regions.

Acknowledgements

First and foremost, I would like to thank my supervisors Prof. Roberto Garello and Dr. Alessandro Compagnoni for their help and support throughout the development and composition of this thesis. I am also deeply thankful to Prof. Juan Fraire for his guidance and his insights during my research work.
I extend my gratitude to Prof. Carla Fabiana Chiasserini, Gabriel Maiolini Capez and Daniel Gaetano Riviello for their excellent feedback on this challenging task.

I want to express my sincere appreciation to my family, for their profound love and belief in me without which I could not have reached this significant milestone. To my dearest Agnese and Valentina, thank you for your friendship, for the laughter, and for the memes in our group chat that keep me company no matter where I am.

Finally, a special thank you to my love Giacomo, for his patience, his unconditional support and for always being by my side during the highs and lows. I am happy to be his girlfriend (even though he did not even thank me in his thesis).

Table of Contents

List of Tables

List of Figures

Acronyms

IoT
Internet of Things.

5G
fifth generation of mobile networks.

3GPP
3rd Generation Party Project.

eMBB
enhanced Mobile Broadband.

URLLC
Ultra Reliable Low Latency Communications.

mMTC
massive Machine Type Communications.

NTN
Non-Terrestrial Network.

XR
Extended Reality.

AR
Augmented Reality.

VR
Virtual Reality.

NET4AI

Network for AI.

NGSO

Non Geo-stationary Satellite Orbit.

VLEO

Very Low Earth Orbit.

ISL

Inter-Satellite Link.

LEO

Low Earth Orbit.

MEO

Medium Earth Orbit.

GEO

Geostationary Earth Orbit.

AI

Artificial Intelligence.

ML

Machine Learning.

RL

Reinforcement Learning.

MIMO

Multiple Input Multiple Output.

UE

User Equipment.

RRM

Radio Resource Management.

ECEF

Earth-Centered Earth-Fixed.

LOS

Line Of Sight.

RAAN

Right Ascension of the Ascending Node.

DoD

Department of Defence.

GPS

Global Positioning System.

COTS

Components-Off-The-Shelf.

NASA

NAtional Aeronautics and Space Administration).

ESA

European Space Agency.

HAP

High Altitude Platform.

UAV

Unmanned Aerial Vehicles.

OFDM

Orthogonal Frequency Division Multiplexing.

ICI

Inter Carrier Interference.

QoS

Quality of Service.

WGS84

World Geodetic System 1984.

A2A

Ascending to Ascending.

A2D

Ascending to Descending.

D2D

Descending to Descending.

D2A

Descending to Ascending.

TAAT

Targeted for Arctic and Antarctic Tracking.

FCC

Federal Communications Commission.

Chapter 1

Introduction

Nowadays the ever-increasing demand for global connectivity and higher performance in terms of coverage, data throughput, reliability and latency acts as a driver for the advancements of communication technologies and the introduction of new standards.
Wireless connectivity is relevant across all industrial fields, though emerging sectors such as telemedicine, precision agriculture, autonomous vehicles, IoT, are particularly challenging and require continuous innovation and development of new mobile broadband frameworks, which include also the integration of satellite constellations to support coverage in remote and under-served areas.

The most relevant mobile network standard used worldwide commercially for wireless communications at the moment is 5G, edited by the 3rd Generation Party project (3GPP). Its main use cases are:

- Enhanced Mobile Broadband (eMBB) whose aim is to guarantee high bit-rate to users (circa 1 Gbps of experienced rate and 10 Gbps of peak-rate).

- Ultra Reliable Low Latency Communications (URLLC) that targets a latency of 1 ms and it is crucial in providing high reliability and quicker response for real-time applications such as remote surgery and autonomous vehicles.

- massive Machine Type Communications (mMTC) that grants Internet connectivity up to one million of low-cost and low-powered sensors per square kilometer.

3GPP is now developing 5G Release 18 and Release 19, the so-called *5G-Advanced* [1], with a collaboration of both the academia and the industry. It guarantees better performance for all the above-mentioned use cases (especially eMBB) and focuses on massive MIMO (Multiple Input Multiple Output) evolution, better positioning for the users, the introduction of AI (Artificial Intelligence) and ML (Machine Learning) and green networks (energy saving by reducing power consumption both at the UE (User Equipment) and the network side). Noteworthy is the discussion of the topological advancements by the integration of NTN (Non Terrestrial Networks) that will be crucial for 6G standard as well: traditionally satellite networks and terrestrial ones have been implemented separately and have dedicated user devices each. The terrestrial network covers only 20% of the land and 6% of the overall earth surface [2], where it provides small latency and optimized routing algorithms. The reason for such low coverage is caused by the potential low return of investment with respect to the sustained costs. Therefore, its infrastructure cannot reach isolated regions with low population density such as rural locations or areas with geographical challenges (for instance, the poles, the deserts and remote islands). With *5G-Advanced* and then 6G, NTN not only will guarantee connectivity in these scenarios but also will increase performance in positioning and navigation of users, ensuring extreme coverage.

Still, in the future, there will be various scenarios that will require even stricter and compelling performance of the next generation of mobile network standards. To give an example, the introduction of new technologies that ensure an immersive experience to the user like Extended Reality (XR), Augmented Reality (AR), Virtual Reality (VR) will necessitate the development of holographic displays and haptic communication (the ability to interact with devices through touch). A direct game-changing application is remote surgery that will allow doctors to save lives without being physically present in the operating room. Other instances can be automation in the Industry 4.0 (motion control in the automation line), smart cities, autonomous vehicles that will substitute completely human drivers, high-precision localization and positioning, the goal of reaching 100% coverage on Earth. All the above-mentioned cases need even higher data-rates (of the order of Tbps), very low latency (of the order of microseconds) and ubiquity with seamless connections.

Building upon *5G-Advanced*, 6G aims at achieving these performances. The world of telecommunications will pass from having connected people and connected things to connected pervasive intelligence. It has been envisioned as the standard

that will make AI the crucial backbone of the network. The latter will not have a cloud-based computing infrastructure anymore, since it will become a distributed learning one (defined as NET4AI). In terms of eMBB 6G offers an experience rate of 10 to 100 Gbps and peak rate of 1 Tbps to users, improves mMTC by having sensing battery life up to 20 years and 10 million sensors per squared kilometer, reaches latency of 0.1 milliseconds, positioning precision of 1 cm indoor and 50 cm outdoor and guarantees a reliability of 99.99999%.[3]

In 6G, due to the rapid technological advancements in the aerospace engineering field and the drop in the launch and manufacturing costs of satellites, it is possible to place thousands of satellites at lower orbits. Thus, the integration of NTN (therefore satellite networks) with the traditional terrestrial one is performed through the employment of distinct layers of Non Geo-stationary Satellite Orbit (NGSO) mega-constellations (satellite vehicles are between 300 km and 2000 km of altitude with respect to the surface of the Earth). Since they are closer to the ground than Geo-stationary Earth Orbit Satellites (GEO) and Medium Earth Orbit (MEO), NGSO satellites and in particular Very Low Earth Orbit (VLEO) ones, have several advantages [4]:

- fewer presence of space debris,
- reduced transmission delay to users on the ground,
- higher signal power,
- lower propagation loss,
- better resolution for real-time Earth observation,
- full coverage of the Earth (even to remote areas),
- back-up in case of critical situations and natural calamities (earthquakes, floods and other extreme environmental events),
- connectivity to users on the move (for example to users on flights or on boats),
- increased performance for navigation and greater accuracy in positioning,
- lower maintenance costs,
- due to the atmospheric drag, satellites can be dismissed more easily when they end their operational period,
- the users will be equipped with terminals that can connect directly to both the cellular network on the ground and the satellites NTN.

Of course, there are various drawbacks as well. For instance, the gravitational perturbations of the Earth, the Moon and the Sun together with the already mentioned atmospheric drag, which is stronger at lower altitudes, can slow the satellite vehicle down and cause an alteration of the satellite's orbit, thus reducing its lifespan in space. NGSO and especially VLEO have a smaller footprint, therefore to cover the same area of a GEO satellite, several VLEO must be well-coordinated. Furthermore, since they are moving at a higher rotational speed, they travel for shorter orbital periods around Earth (generally in 1.5-2 hours). As a result, the topology of the network dynamically changes. Special attention must be given to this last concept because the faster the satellites are, the more frequent the handovers will be for ground stations or users that try to connect to the ones in their line of sight.

Fast topology changes and the management of data sent along Inter-Satellite Links (ISL) are also the fundamental drivers for research advancements in the development of new routing techniques for the commonly named *space segment* (thus the section of the satellite network that comprehends the paths from the source and destination satellites, that are supposed to be selected by the users on the ground and known by the network before performing routing).
Moreover, mega-constellations imply huge routing tables. Consequently, there is the open issue of where the routing tables should be stored and how often routing should be performed, choosing between offline solutions and on-demand ones. For instance, in multi-layer layer constellations NGSO satellites are coordinated by higher layer (in general, MEO) ones that compute the paths and keep track of the overall network state.

Routing can be studied deterministically, after building up constellation models such as Walker Star and Walker Delta. The orbit of each satellite is known, thus it is possible to make predictions on where the satellite and its neighbours will be to make routing decisions. In literature, the neighbours of the satellite under test usually are four: two in the same orbit, immediately before and after it and two in the adjacent orbits.
Another strategy used to implement routing is to handle frequent network topology shifts with ML. The latter is a solution that is not based on deterministic modeling but on interpreting the received data defining a scheme that fits it and using it to forecast results. Each satellite node learns the forwarding policy independently from the others. For instance, when a link fails the node can straight away re-route traffic on another path. The most utilized technique by nodes to acquire knowledge of the network is *Q-learning* that is a branch of Reinforcement Learning (RL).

In the following master's thesis project, the performance of different routing techniques will be examined for each satellite pair of a constellation under study. The term "pair" refers to the couple *source satellite* and *destination satellite*. The analysis is provided first on aLEO (Low Earth Orbit) constellation and then extended on a VLEO (Very Low Earth Orbit) one. The research work will focus particularly on `DisCoRoute` algorithm [5], designed for Walker Delta LEO constellations. After verifying the results of the paper in terms of propagation delay and computational time, the evaluation is expanded to a VLEO configuration. Results highlight a drop in performance of `DisCoRoute` around the polar regions. A further analysis is carried out to investigate the relationship of the routing performance to the different altitude, inclination and topology (number of planes and number of satellites per plane). Finally, a new heuristic is proposed to handle the issue in the Arctic and Antarctic areas.

The thesis is organized as follows:

- In chapter 2, corresponding to the background, a description of the main concepts useful to fully comprehend the thesis is provided.
- In chapter 3 the state-of-the-art is presented as a collection of the most recent methods, techniques and innovations that are available nowadays regarding LEO and VLEO satellites and routing.
- In chapter 4 the overall structure of the study is shown in terms of design choices and selected algorithms, showing the detailed description of each step of the procedures used to analyze the data and generate the results, both taking into account the outcomes and granting their reproducibility.
  The practical implementation of the illustrated techniques is displayed as well.
- In chapter 5 findings of the above-mentioned analysis are shown, explained and interpreted, including a comparison with existent papers, taking into consideration the equipment and the software employed.
- In chapter 6 as a conclusion, a synopsis of the work is delineated, restating the procedures with the obtained outcomes and highlighting the strengths and the limitations of the utilized approaches. Future research implications are discussed as well.

Chapter 2

Background

2.1 Overview

This section aims to introduce all the fundamental concepts necessary for understanding the project and its implementation. It begins by explaining what a satellite is and its orbital characteristics, providing an in-depth exploration of Non-Terrestrial Networks (NTN). Following this, it presents an initial definition of satellite constellations along with their essential parameters. Lastly, it covers basic concepts of routing and the routing strategies that will be employed throughout the thesis.

2.2 Satellites and their orbital characteristics

Conventionally, a satellite is an object that revolves around another of a larger size in a periodical fashion. It is classified as either *natural* or *artificial*. A *natural* satellite is a celestial body that rotates around a planet. For instance, the Moon orbits around the Earth, thus being known as the only natural satellite of the Earth.

The research work of the thesis will focus on *artificial* satellites. As a definition, an *artificial* satellite is a man-made vehicle sent up into space to travel an elliptical and periodical trajectory around Earth. Its central scope is to be used to collect and deliver information from either the Earth or space and send it back either to the ground or to other satellites.

When discussing satellite systems, it is crucial to specify the reference system with respect to which the satellite's position, speed, and other characteristics are defined. For instance, in the following study, two reference frames will be frequently used. The Earth-Centered Earth-Fixed (ECEF) one corresponds to a Cartesian coordinate system in which the origin is the center of the Earth, with x and y that are on the equatorial plane. Noteworthy is the fact that it rotates with Earth. The other one is the Geodetic Coordinate Frame, which employs latitude, longitude and height coordinates to identify both satellites in space and users on the surface of Earth.

Taking into account the point of view of an observer on the ground, a satellite is detected in the sky depending on a few specifications, such as, first of all, the reference system. In this scenario, the one employed is ECEF. In addition, other key parameters are the *elevation*, the *zenith* and the *azimuth* [6]. The above-mentioned considerations are frequently utilized when a user wants to compute its own position. It is a process called *trilateration*: the user understands its own location on the ground calculating the distances with respect to at least four satellites in its horizon of visibility.

As shown in Figure 2.1:

- the *zenith* is the point in the celestial spherical cap on top of the observer.
- the *horizon* is the plane tangent to the Earth's surface at the observer's position. It is necessary to determine which satellites are in visibility of the observer.
- the *elevation* is the angle between the straight line joining the observer's and satellite's respective locations and the horizon plane. It goes from 0° to 90° when the satellite is at the zenith. Habitually a satellite is intended to be "in Line Of Sight (LOS)" when its elevation angle is at least above 5°.
- the *azimuth* is the angle measured on the horizontal plane between the direction from the observer to the North and the projection on the same plane of the straight line joining the observer's and satellite's respective locations.

The location in space of a specific orbit and the position of each satellite within the orbit are identified by six criteria, known as *keplerian orbital parameters* [7],

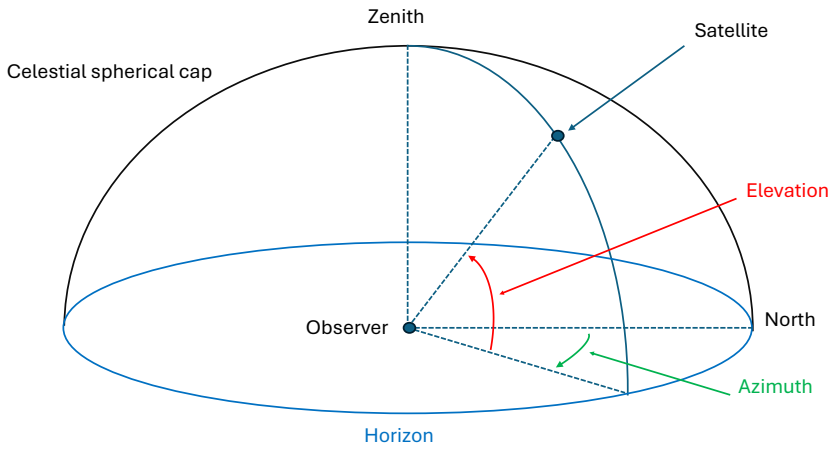


Figure 2.1: Satellite's specifications from observer's point of view

that can be converted into geodetic and cartesian coordinates. They are illustrated in Figure 2.2:

- the *inclination* (α) is the angle that the orbital plane generates with respect to the equatorial plane, measured at the *ascending node* (that is the point in which the satellite crosses the equatorial plane while ascending, meaning it goes from the south hemisphere to the north one). It is worth mentioning that the equatorial plane itself has a sinusoidal variation of its own inclination with respect to the straight line connecting the center of the Earth and the Sun over a period of 365 days (during which it is possible to determine solstices and equinoxes).

- the *eccentricity* (e) is a value that indicates how much a curve is closer to a circumference. It can be classified as follows:

$$e = \frac{apogee - perigee}{apogee + perigee} = \begin{cases} 0 & \text{circumference} \\ (0,1) & \text{ellipse} \\ 1 & \text{parabola} \\ > 1 & \text{hyperbole} \end{cases} \tag{2.1}$$

  where the *apogee* is the point in the orbit where a satellite is at maximum distance from the center of the Earth and the *perigee* where it is at minimum.

- the *semi-major axis* (a) of the orbit is the sum of periapsis and apoapsis divided by 2.
- the *argument of periapsis* is the angle between the line joining the center of Earth with the ascending node and the one connecting the center of Earth with the perigee.
- the *true anomaly* (ω) is the angle between the line from the focus of the ellipse (in case of a circumference, the center of the Earth) to the perigee and the line joining the center with the satellite.
- the *Right Ascension of the Ascending Node* ($RAAN$) (Ω) is the angle in the equatorial plane from the direction taken as a reference (the vernal equinox) and the one from the center of the Earth pointing at the ascending node.

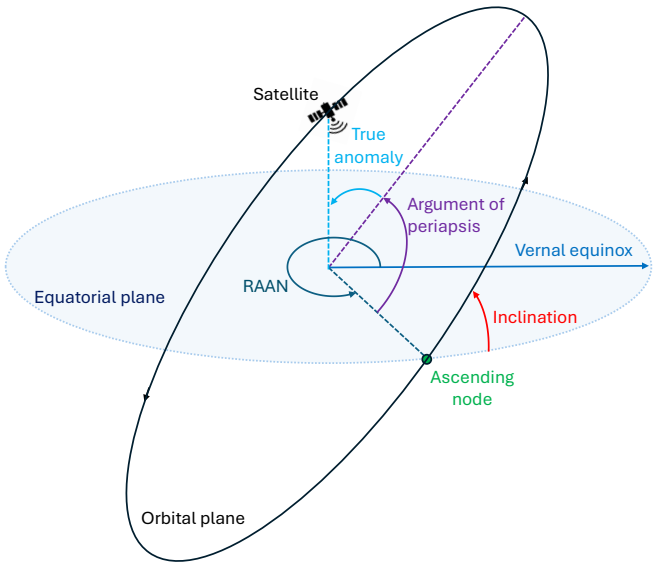


Figure 2.2: Keplerian orbital parameters

The satellite vehicles analyzed in the thesis travel a circular orbit around Earth, thus being a scenario whose behaviour can be modeled as a *two-body problem*. Indeed, the latter is used to describe the situation in which an object moves with respect to another, given that the two only interact with each other. Newton's law of universal gravitation and Kepler's laws of planetary motion, which are illustrated concisely in the following paragraphs, are the fundamental pillars that characterize such model.

Newton's gravitational law is given by equation 2.2:

$$\mathbf{F} = G\frac{m_1 m_2}{r^2}\hat{\mathbf{r}} \tag{2.2}$$

It means that the magnitude of the gravitational force vector $\mathbf{F}$ between two masses is directly proportional to their product and inversely to the square of the distance of their center of mass, whereas $\hat{\mathbf{r}}$ is the unit vector starting from one mass to the other. G is the universal gravity constant:

$$G \approx 6.67 \times 10^{-11}\,\mathrm{m^3 kg^{-1} s^{-2}}$$

Kepler's first law of planetary motion is: *"The orbit of a planet (satellite) around the Sun (Earth) is elliptical, with the Sun (Earth) being one of the two foci"*. In the cases of LEO and VLEO constellations, that constitute the pivot of the current research work, as already mentioned the orbits are circumferences (e = 0), thus implying that the center of the Earth coincides with the center of the orbit.
Next, Kepler's second law is: *"The imaginary line between the planet(satellite) and the Sun(Earth) sweeps out equal areas in equal amount of time"* suggesting that in a circular orbit the speed of the satellite is constant. Finally, Kepler's third law is: *"The square of the orbital period of any satellite is proportional to the cube of the semi-major axis of its elliptical orbit"* [7].

In order to comprehend the subsequent contents, it is necessary to make a brief digression and introduce also the concepts of *propagation delay*, *horizon time* and *antenna footprint* [8].

- The *propagation delay* is the time t needed by a signal to travel at the speed of light c through the distance d between a transmitter and a receiver. It is computed as $t = \frac{d}{c}$ where $c \approx 3 \times 10^8$m/s. In the context of satellite networks, both the transmitter and the receiver can be either a satellite or a ground station/user on the surface of the Earth.

- The *horizon time* is the interval during which a satellite is in the visibility region of the observer from the ground.

- The *antenna footprint* is the geographical area illuminated by the antenna's beam of a specific satellite. It delineates the region within which users can receive the satellite's signal and vice versa.

When referring to satellite communication systems, it is quite common to separate the *space segment* (that comprehends both the space stations and the satellite infrastructure in space for earth observation, communication and navigation) from the *ground segment* (which encompasses the user terminals, the mission control

centers and the ground stations). Focusing on the *space segment*, a satellite can belong to GEO, MEO, LEO and VLEO categories, depending on the altitude with respect to the ground:

- GEO satellites orbit at 35786 km above Earth's surface. They travel at the same rotational speed of the Earth with the same 24-hour period, therefore when an observer on the ground looks for them, they can be found at a fixed position in the sky. GEO satellites are traditionally employed for broadcasting services and weather monitoring. On one hand, an interesting benefit of using GEO satellites is the coverage stability, which is continuous over a specific region (they have an antenna footprint of 1000 km square). On the other hand, an obvious drawback is that the long distance from the surface of the Earth causes a significant propagation delay (circa 270 ms) which impairs the applications of GEO satellites for real-time or emergency scenarios.
- MEO satellites operate in the space region between 2000 km to 35786 km. Their period is 12 hours, thus they can be seen from the ground twice a day for a horizon time of maximum 2 hours. Their antenna's footprint is circa 500 km and has a latency of 94 ms.[8]. They are fundamentally used for positioning and navigation purposes, both for military and civilian applications. For instance, the DoD of the USA launched the GPS (Global Positioning System) in the 1970s which is composed of 24 MEO satellites at 20000 km (4 distinct satellites in each of 6 orbital planes).
- LEO satellites are placed between 500 km and 2000 km. Their period is 2 hours and the horizon time is 10 minutes. The latency is lower (circa 20ms) and the antenna's footprint is 100 km squared. Since they are at a lower altitude with respect to the Earth's surface, to escape the gravitational pull and in order to maintain their orbit, their rotational speed is very high. Furthermore, to cover the same geographical region as Geo Stationary Orbit satellites, many more satellites need to be coordinated.
- VLEO satellites travel through orbits below 500 km in periods of at most 90 minutes. On the one hand, they guarantee even better latency than LEO (about 5 ms) and very good image resolution. Besides that, at very low orbits there is less risk of encountering space debris and colliding with other satellites. On the other hand, the lifespan is shorter due to the atmospheric drag and radiation effects.

2.3 Overview on Non-Terrestrial Networks

Big private companies both in the US and Europe such as SpaceX (Starlink), Project Kuiper (Amazon), OneWeb and Iridium Communications have been investing

consistently for years in aerospace and communications technologies to extend coverage to unconnected regions, providing low latency and high data throughput to users even in emergency situations. Their strategic pivot is the provision of NTN services thanks to the production, launch and coordination of thousands of LEO and VLEO satellites (creating the so-called *mega-constellations)*. Since they are produced with Commercial-Off-The-Shelf (COTS) components, NGSO satellites are not as expensive as GEO and MEO ones. The latter types of satellites are still manufactured and controlled by government space agencies (namely, NASA and ESA).

Non-Terrestrial Networks are infrastructures built up either on air (defined as *Air-borne platforms*) or in space (also called *Space borne platforms*) [8]. Their main goal is to reach global coverage by offering connectivity not only in remote areas with low density population but also in harsh environments and disaster scenarios. However, they are also decisive in supplying even better performance in navigation and positioning, paving the way for very useful applications like precision farming and autonomous vehicles. Fundamental will be the integration in the 6G standard of NTN with existing terrestrial networks as well.
In Figure 2.3 an overall depiction of NTNs is presented.

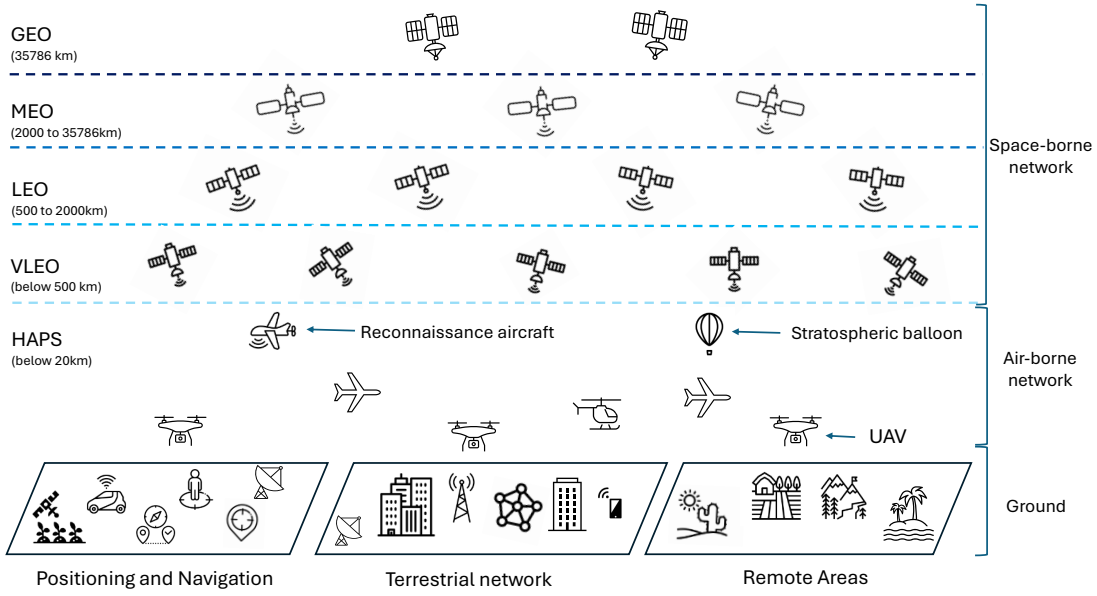


Figure 2.3: Non-Terrestrial Networks

Air-borne networks are a broad ensemble of network configurations generated at altitudes lower than 22 km that are meant to guarantee coverage with low

propagation delay to users in disaster scenarios and remote areas. They can be designed either as a collection of scientific research balloons or reconnaissance aircraft that generally operate at between 17 and 22 km of height with respect to the ground (in the stratosphere), forming the High Altitude Platform (HAP) or at lower altitudes (around 10 km) with air-crafts (planes, airships) and Unmanned Aerial Vehicles UAV (drones). The limitations in the use of air-borne networks are that they have a restricted antenna footprint and present challenges in the stability of the routes and in the robustness of components, constantly stressed by hostile weather conditions and adverse temperatures. The high costs of maintenance are a drawback as well. Therefore, the focus of the current research work has been the establishment of satellite constellations that can reach global coverage and overcome weather sensitivity.

Space-borne networks are composed of collections of satellites at different altitudes, each forming a distinct layer (or shell). To give an example, the farthest layer from Earth comprehends only GEO satellites, whereas the closest is made of VLEO ones. The layers may or may not interact with each other in case of a specific task to achieve (for instance, in some routing strategies the routing decisions can be computed by MEO satellites and sent to the LEO ones). Commonly, the more distant the layer, the more costly for a company.
In this type of networks, either the ground stations or directly the users (UE) connect to the satellites that are in their line of sight for a specific period of time that is called the *horizon interval*. Such satellites are known as *access satellites*. They can act as relays between users and also regenerate their signal, otherwise they are able to communicate with other satellites both in the same and other shells through ISLs.
The most utilized frequency bands to provide high speed Internet connectivity are:

- S-band: from 2 to 4 GHz
- Ku-band: from 12 to 18 GHz
- Ka-band: from 26 to 40 GHz

As already mentioned in chapter 1, NGSO satellites take approximately 2 hours to conclude an orbit around the Earth because they travel at much higher rotational speed than that of the Earth. Given this velocity, the ground observers (being them either users or ground stations) can connect to an access satellite for a limited amount of time, usually just a few minutes, before the satellite moves away from their visibility space. This situation implies that to avoid any connectivity disruption, UEs must switch and link to other satellites in a short interval (thus having to undergo multiple handovers).

Moreover, they have also to consider and compensate for the effect of the Doppler shift in the received/sent signals, due to the relative motion of the satellite with respect to them. To give an example, suppose a satellite is moving along its own orbit and transmitting a signal to a ground station. As soon as it advances toward the ground station, the latter will receive a signal whose carrier frequency will be higher than the transmitted one. On the contrary, when the satellite distances itself from the ground station, the received signal will have a carrier frequency that is lower than the one of the transmitted signal. In equation 2.3, the above-mentioned situation is mathematically described [9]:

$$f_{rx} \begin{cases} f_{tx} + f_{Doppler} & \text{satellite moving forward the receiver} \\ f_{tx} - f_{Doppler} & \text{satellite moving away from the receiver} \end{cases} \tag{2.3}$$

where the Doppler shift, given the speed of light c and the speed of the satellite v (relative to the ground station), is given from equation 2.4:

$$f_{Doppler} = f_0 \frac{v}{c} \tag{2.4}$$

In substance, if not handled, the Doppler effect produces frequency offsets (shifts) that cause misalignments and loss of synchronization at the receiver. As an example, it is well known that OFDM modulation systems suffer from inter-carrier interference (ICI) when affected by multi-path channels exhibiting different Doppler shifts [10].
Noteworthy is the fact that in 6G the integration of NTN with the traditional network will be done even with the design of new UEs able to exchange information with both the traditional terrestrial networks systems and different NTNs, thus substituting a collection of distinct devices, each dedicated to a single type of network.

2.4 Portrayal of Satellite Constellations

A *satellite constellation* is a collection of artificial satellite vehicles that travel either circular or elliptical trajectories around Earth. Each satellite works with the others in a coordinated fashion to perform a specific task, thus forming a true infrastructure in space. All constellations have the main goal to provide global coverage to users on the surface of the Earth, by connecting directly to them or through ground stations allocated around the planet.
As a rule of thumb, a constellation is composed of a fixed number of orbits, each of them with a constant number of equally spaced satellites [11]. If the orbits are circular and have the same inclination, with satellites at the same altitude from the ground, then the just mentioned scenario is classified as an *orbital shell*.

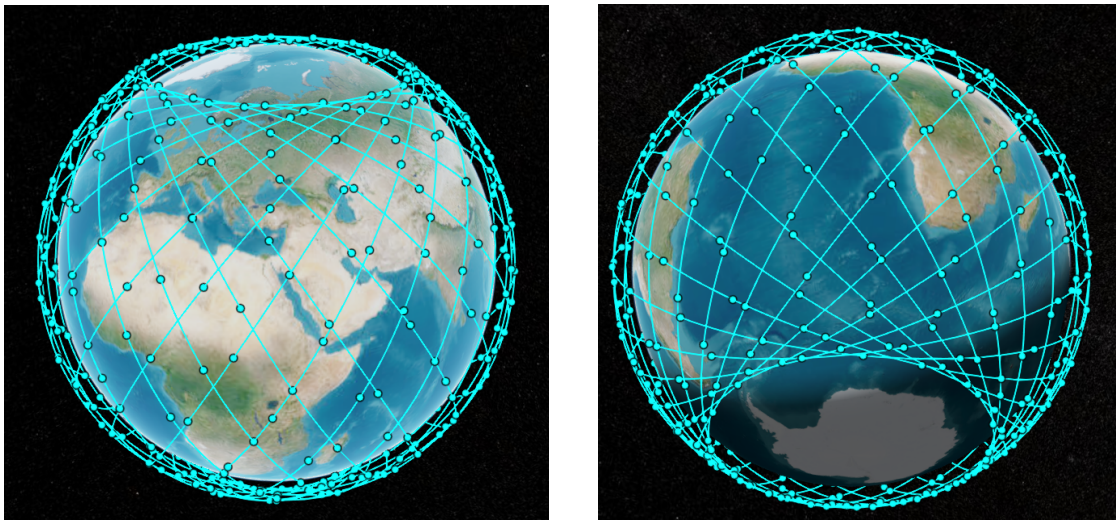
Mega-constellations (which comprehend thousands of satellites, generally either LEO or VLEO) may also be composed of several orbital shells communicating with each other in a hierarchical pattern, for example for routing purposes.

The design of NGSO constellations in literature has been defined through two models, the *Walker Delta* and the *Walker Star* ones, proposed by engineers J. Walker [11] and A. H. Ballard in 1985 [12].
Walker constellations are described by a well-known notation which is α:T/P/F where:

- α represents the inclination of the orbital planes with respect to the equatorial one.
- P is the number of orbital planes of the constellation.
- T is the total number of the satellite constellation, therefore it is the product of the number of planes P with the number of satellites per plane Q and can be also denoted as PQ.
- F is the phasing factor, it is a parameter that gives information about the spacing between satellites in adjacent orbital planes. Moreover, it assumes only integer values that belong to the interval $[0, P-1]$.

As already mentioned, orbits are also defined as circular. For this reason, the eccentricity is always null and the speed of the satellite is constant (for example in case of LEO it is around 7.5 m/s). Additionally, the semi-major axis is always equal to the radius, and there is not a specific point where it is possible to identify either the apogee or the perigee.
In particular, the absence of the perigee causes a problem when delineating the true anomaly. As a solution, the latter is not used anymore and is substituted by a new quantity, called *argument of latitude*. It is the angle between the line connecting the center of the Earth and the ascending node and the one linking again the center with the position of the satellite under test in the orbit. When dealing with the Walker layout, the *initial longitude of the ascending node* constitutes another key parameter. In the context of this thesis project, its value is set to zero.

The Walker Delta configuration is a set of orbits arranged in a flower-like pattern (that is the reason why it is also referred to as the *Ballard-Rosette* model). The orbits are circular and allocated to different planes evenly spaced within 360° along the Equator. Moreover, they have the same inclination (generally below 60°) and the same altitude from the surface of the Earth to avoid variations in the transmitted signal power. Satellites are evenly distributed in each orbit, such that they don't collide with each other in orbital plane intersections at the poles.

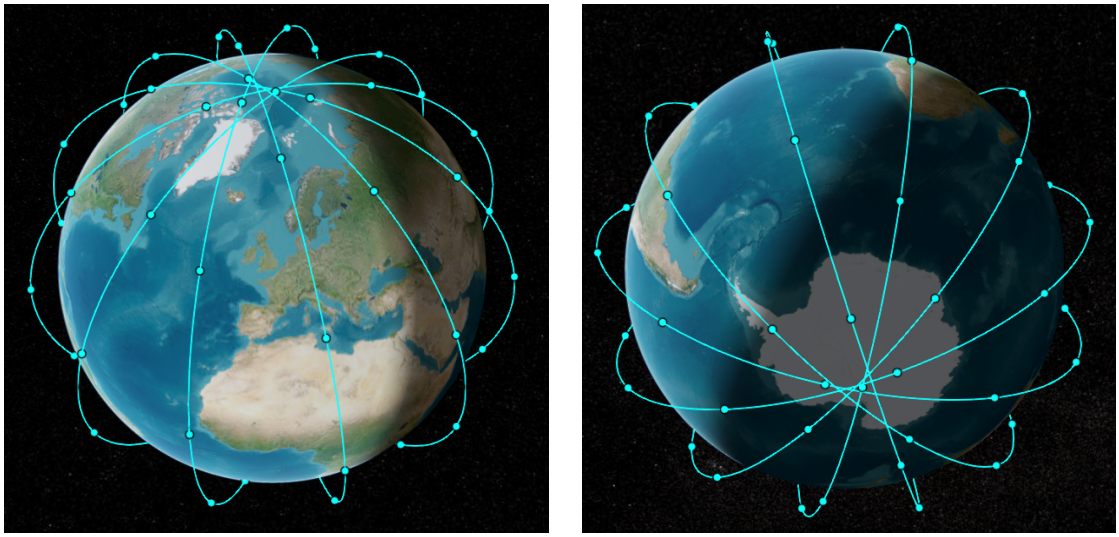


(a) LEO constellation 60°:500/25/5 (1) **(b)** LEO constellation 60°:500/25/5 (2)

Figure 2.4: Walker Delta configuration for LEO

To give an example, Galileo constellation, which is the official satellite infrastructure built up by several European countries and under the control of ESA, is a Walker Delta pattern of MEO satellites at around 23000 km. It is represented by the notation 56°:27/3/1 [13] It means that there are a total of 27 satellites distributed along 3 orbital planes (9 per plane). Satellites in adjacent orbits are aligned and have the same latitude since the phasing factor is 1.
The vast majority of LEO and VLEO constellations are based on this described model, such as the ones from private companies like Starlink. In Figure 2.4 a plot of the Walker Delta pattern for LEO constellation 60°:500/25/5 is shown from two different angles (Figures 2.4a and 2.4b). There are 500 satellites with phasing factor equal to 5 that are distributed along 25 planes. The latter ones are all inclined by 60° with respect to the equatorial plane. Noteworthy is the fact that with this type of configuration, no satellite covers the full Arctic and Antarctic regions.

The Walker Star pattern uses the same Walker notation. Typically, the inclination is higher than the Walker Delta (therefore, around 90°). The big difference is the disposition of orbital planes and the corresponding ascending nodes on the Equator. They are equally placed around 180° and not 360°. It is a configuration mainly used by Iridium company.
The most known layout from Iridium is represented by notation 86.4°:66/6/2 at 781 km orbit altitude. It means that there are 66 satellites placed in 6 different orbital planes with same inclination 86.4° and phasing factor equal to 2 [14]. Its depiction is presented in Figure 2.5 from two different viewpoints as well (Figures



(a) Iridium constellation 86.4:66/6/2 (1) **(b)** Iridium constellation 86.4:66/6/2 (2)

Figure 2.5: Walker Star configuration for LEO

2.5a and 2.5b). As a comparison with the previous arrangement of satellites, the Walker Star setup can actually provide full coverage to the polar areas.
In the following thesis, all the models analyzed both for LEO and VLEO will focus on the Walker Delta configuration since it is the most utilized one.

Focusing on the type of links in a satellite constellation, in both the Walker layouts, each satellite is connected to the others through 4 links, 2 intra-orbit (within the same orbit) and 2 inter-orbit (between adjacent orbits), also called *Inter Satellite Link* (ISL).
In the case of *multi-layer constellations*, which are space network systems composed of more than one orbital shell (for instance, one layer is made by GEO satellites, one by MEO and one by LEO) there are also the inter-layer links.

2.5 Routing Strategies

In traditional communication networks, *routing* is a set of strategies, decisions and algorithms that allows to determine the best path to connect a source node to a destination one. The goal is to send information through packets that can have either fixed or variable size in the most efficient way possible. The most suited path is always delineated with respect to a precise metric. For instance, the best path may be the shortest one, however as an alternative it could also be the less

congested one or the path that guarantees some specific Quality of Service (QoS) requirements (like reducing packet loss, bandwidth allocation and giving priority to a certain traffic category in respect of others).
In the context of NTN, the satellites of a constellation are considered as nodes of a topology, with the links between them that can be weighted or not with a metric. Therefore, both academia and private companies worldwide are looking for new methods to transfer and adapt the know-how of routing of terrestrial networks in space. Of course, satellite constellations present a challenging and dynamic environment with different traffic requirements. Besides that, satellites have restricted on-board storage space. The need for the implementation of new routing algorithms is clearly compelling.
Routing tables and other types of delays besides the propagation one become as crucial in satellite communication networks as well as they are in terrestrial ones [15].

- The *routing table* is a logical data structure stored in satellite nodes, that contains details about the source node, the destination, the selected metric, the port where to send data and the next hop. The latter is computed with an ad-hoc algorithm.
- The *processing delay* is the time required by the satellite to process the header of a packet and select the most suited routing choice.
- The *transmission delay* is the time that a satellite employs to send a packet. It is equal to the ratio between the packet size and the data transmission speed.
- The *queuing delay* is the time a packet has to wait inside the buffer of a satellite node before being transmitted.
- The *end-to-end delay* between the source and destination satellites is the sum of the aforementioned delays caused by each intermediate node, along with the propagation delay.

In general, routing in a satellite communication network system can be tackled by three different approaches that are called *boundary routing*, *access routing* and the most important one, *on-board routing* [16].
Both boundary and access routing address the portion of the network infrastructure between the ground segment and the access satellites (which act as intermediaries with the space segment). Essentially, boundary routing protocols' main aim is to allow the interoperability between the NTN and the terrestrial networks, whereas the access routing examines the implications of connecting to one specific access satellite with respect to the others.
On-boarding routing is currently the field where most of the research is concentrated on since it looks for the best path between a selected source satellite and a

destination one. It involves only the space segment.

Routing algorithms can also be tailored based on the type of constellation layer they are aimed at. If it is a single-layered one (therefore, only one orbital shell) then the routing can be done based on either a *virtual topology* or a *virtual node*. All the satellites belonging to the same layer move periodically, such that after a certain amount of time, they come back to their original position. In the *virtual topology* strategy, which is the most utilized line of action, snapshots of the constellation are captured at regular small time intervals, so that the constellation can be considered fixed and routing is performed within that duration. What actually changes between different time spans is the disposition of the links between satellites. In the *virtual node* approach, the satellites move, but their logical position remains constant, so that when a satellite leaves its own location, that same spot is occupied by another satellite of the same orbit. In this case, routing algorithms can be carried out on the obtained fixed logical topology, independent from satellites' motion.
Alternatively, regarding the multi-layered structure, the constellation is arranged in either 2 or 3 orbital shells performing a master-slave scheme with the help of inter-layer links. The most common scenario is the one where there is one MEO satellite that acts as a master. It collects information about the state of the network and computes the routing tables and best paths for the LEOs that take the role of access satellites for the ground users. Then, the master sends the routing decisions to the MEOs responsible for the LEO satellites in a definite area. The MEOs sends only the routing entries to the LEOs. Finally, the LEO satellite source can start sending packets over the selected path.

Routing algorithms can also be categorized making a distinction between *centralized* and *distributed routing fashion*.
Centralized routing can be applied when the topology of the network is not changing frequently. In substance, there is a control entity, which has the complete knowledge of the topology of either the entire or a significant portion of the network. It is able both to compute the routing tables and to make routing decisions for every satellite in that same area. In this way, it communicates the path to follow to each node, thus saving computational time and end-to-end delay for each satellite and reducing network management issues. The main drawback is that if there is a problem in the control entity, it causes an impact on the entire section of the network under its supervision.
Instead, in a distributed scheme, each satellite computes its own routing table, based on the information gathered by communicating with its neighbours. This approach has the advantages of gaining adaptability and flexibility when network changes happen. Unfortunately, it sometimes produces considerable overhead in

packets and scalability problems which increase the network management complexity.

Finally, a last classification is made by distinguishing between a deterministic approach and one that uses ML.
In the first setting, routing algorithms are implemented so that, once the inputs and the initial conditions of the scenario are defined, the outcome of the procedure is optimized and immutable for that particular case. The procedure itself is predictable and fixed since it is based on a mathematical model in the methodology. For instance, in [17] and [5], the constellation is set out as a Walker Delta. Then the on-board routing is computed, in a distributed fashion, by considering a mathematical model about the minimum number of hops and the directions that packets should travel through from the source satellite to the destination satellite. The just mentioned obtained minimum hop count reduces the search space to a grid where two vertices are the source and the destination and it is always possible to reach the destination through the same number of hops with the Manhattan distance. Essentially, [5] introduces a new sub-optimal algorithm called `DisCoRoute` and compares its performance in terms of computational time and end-to-end delay with the other well-known algorithms (such as the optimized single-pair shortest path `Dijkstra` routing algorithm) [18].

As an alternative, a branch of ML named Reinforcement Learning (RL) is frequently utilized to face the rapid topology changes of the satellite network. The goal of RL is to learn a strategy in a rapidly mutable environment through taking actions and maximizing a cumulative reward. Unlike deterministic models, RL interprets collected data to develop and apply a scheme that adapts to current conditions of the satellite network and forecasts future outcomes. In this approach, each satellite node independently learns its forwarding policy. For example, if a link fails, the node can immediately reroute traffic through an alternative path. A widely used technique for satellite nodes to learn about the network is Q-learning [19]. The latter is a technology where satellites are considered as agents taking actions and obtaining rewards governed by Q-tables and the Bellman equation.

Chapter 3

State of the Art

3.1 Overview

In this master's thesis, a set of algorithms regarding routing and their application to both Low Earth Orbit (LEO) and Very Low Earth Orbit (VLEO) constellations will be illustrated and explained.
The state of the art is presented as a collection of papers that show the innovations and techniques that are available nowadays in academia regarding LEO and VLEO satellites and routing. At the beginning, a description of VLEO advantages and disadvantages is provided, followed by a set of articles explaining deterministic routing over satellite constellations. Finally, a wide insight on ML and its adaptability to space networks is outlined.

3.2 VLEO benefits and challenges

The work *Very Low Earth Orbit Mission Concepts for Earth Observation - Benefits and Challenges* by Josep Virgili-Llop et al.[20] shows the advantages and limitations of a satellite vehicle in a VLEO constellation (with orbits below 450 km) for Earth

Observation tasks. On the one hand, VLEO offers crucial benefits such as an increased resolution, an improved geospatial accuracy and a decreased payload weight. On the other hand, the article highlights the challenges posed by the atmospheric drag and oxygen erosion, which require new design solutions such as atmosphere-breathing propulsion. The study also discusses the opportunities to reduce satellite size and costs, making Earth Observation missions more flexible and less expensive. Furthermore, VLEO offers natural debris mitigation, allowing satellites to end their operating period without requiring additional costs.

Are Very Low Earth Orbit Satellites a Solution for Tomorrow's Telecommunication Needs? by Lucy Berthoud et al.[21] is an article that explores VLEO satellites as a promising solution when responding to the increasing telecommunication demands. VLEO satellites provide significant benefits like reduced latency, improved link budgets, and lower transmit power. The paper describes the introduction of a VLEO constellation able to guarantee 5G connectivity to 95% of the Earth's population. Each satellite is designed to have 320 beams, providing an average data rate of 3.8 Mbps per beam. The flexibility of the system allows to vary data rates depending on usage. Despite the benefits, the disadvantages already mentioned in the previous paper are discussed, with possible solutions like electric propulsion as compensation for the atmospheric drag.

Another comprehensive review of the advantages of using VLEO for Earth Observation missions is detailed in *The Benefits of Very Low Earth Orbit for Earth Observation Missions* by Nicholas H. Crisp et al.[22]. The main asset of VLEO emphasized by the article is the enhanced spatial resolution for optical, radar, and lidar systems due to the reduced distance to the Earth. VLEO also guarantees higher launch mass efficiency. However, the paper identifies the drawback of the reduced orbital lifetime, which requires further research in propulsion systems and novel materials. The review also draws attention to the growing commercial and environmental importance of Earth Observation tasks, urging continuous innovation and investments in enabling VLEO technology.

The paper *Very Low Earth Orbit Constellations for Earth Observation* by Nicholas H. Crisp et al.[23] examines the trade-offs in system design for deploying VLEO satellite constellations. The increased atmospheric drag is compensated by an advanced propulsion system called "Atmosphere-Breathing Electric propulsion" (ABEP). The study stresses that VLEO constellation configurations need more satellites to achieve global coverage due to the limited field of view at lower altitudes. However, the benefits of improved data resolution and cost-efficiency outperform the just-mentioned issue.

Finally, *System Modelling of Very Low Earth Orbit Satellites for Earth Observation* by Nicholas H. Crisp et al. [24] presents a detailed system modeling framework for designing satellites operating in VLEO. The paper emphasizes the potential of VLEO to reduce satellite mass and manufacturing costs while maintaining or improving its performances. New technologies are designed to understand how they can support the satellite functionalities in the denser atmospheric environment of VLEO. The case studies show up to 75% mass reduction and over 50% cost savings for optical payloads. However, in this paper the challenges of increased atmospheric drag and the need for continuous technological advancements in propulsion systems and materials are pointed out as well.

3.3 Deterministic Routing in Space Networks

In deterministic routing, once the configuration of the constellation is set up, the algorithms select choices that are made based on a mathematical model, thus being specific and immutable for that scenario only and respecting several predefined conditions in the methodology.

A Survey of Routing Techniques for Satellite Networks by Q. Xiaogang et al. [16] provides a comprehensive overview of routing strategies in satellite networks, focusing on both single-layer and multi-layer constellations.
For single-layer networks, two crucial approaches are presented: virtual node strategy and virtual topology strategy. In the multi-layer case, the paper provides insights regarding the inter-satellite links (ISLs) that change frequently between LEO and MEO satellites. The survey also underlines the challenge of implementing good routing algorithms in satellite networks due to limited onboard processing capabilities, high error rates, and long delays. The authors predict future research will focus on improving adaptability to traffic changes, ensuring robustness against network failures, and optimizing routing algorithms for both performance and resilience.

An Overview of Low Earth Orbit Satellite Routing Algorithms by C. Li [25] reviews several routing algorithms implemented for LEO networks, focusing on the unique challenges posed by dynamic topologies and limited resources. Several approaches are explored, including static and dynamic routing, with a particular focus on the use of time-varying graphs for dynamic path optimization. Software-defined networking (SDN) is identified as a promising solution, enabling centralized control and topology updates in real-time. The authors highlight the trade-offs between delay reduction, bandwidth management, and security.

Time-Varying Topology Model for Dynamic Routing in LEO Satellite Constellation Networks by Z. Han et al. [26] describes a time-varying topology model to enable dynamic routing in LEO satellite constellation networks. The model uses a weighted time vs space evolution graph to represent ISLs with dynamic attributes such as signal to noise ratio, link duration, and buffer queue, thus quantifying link utility for routing. The proposed Inter-Satellite Link Utility-based Dynamic Routing (IUDR) algorithm selects routes with the highest link utility, dynamically adapting to topology changes. Simulation results show how the introduced procedure outperforms static routing in reducing packet drop rate, improving end-to-end delay, and enhancing throughput.

The paper *Topological Design and Routing for Low-Earth Orbit Satellite Networks* by H. S. Chang et al. [27] details the joint optimization of both the topology definition and routing for LEO constellations. The authors model the satellite network as a Finite State Automaton (FSA), where satellite visibility and inter-satellite link (ISL) connectivity change dynamically. Then, the joint optimization issue is divided into two problems: link assignment and traffic routing, with the goal of maximizing network performance by balancing traffic loads. A heuristic algorithm is introduced to find sub-optimal solutions for both problems. The previously calculated link and routing tables are stored on satellites and updated during state transitions. Simulation results show that this procedure effectively increases network performance.

Finally, *Performance Comparison of Static and Dynamic Routing in Low-Earth Orbit Satellite Networks* [28] compares the performance of static and dynamic routing schemes in LEO satellite networks using again a Finite State Automaton (FSA) model. Static routing pre-computes routing tables for each state of the network topology, while dynamic routing updates routes in real time using the shortest path algorithm. Simulation results show that static routing outperforms dynamic routing in terms of initiated call blocking and ongoing call reliability. This is because static routing avoids the overhead of frequent updates and performs better during state transitions, whereas dynamic routing struggles to adapt quickly enough to topology changes, resulting in higher call drops.

3.4 Machine Learning for Routing in Space Networks

Machine Learning (ML) fundamental functioning is to collect data, to fit them into a model, and being able to make predictions and forecasts. Its application to the *space segment* is relative to the possible quick changes of the satellite network topology. The key insight is that its employment allows to have a strategy that adapts to constellations that can vary dynamically. Indeed, frequent link failures can be handled rapidly by rerouting traffic.
As previously mentioned, a significant area within Machine Learning (ML) is Reinforcement Learning (RL). In the context of space networks, Q-learning emerges as one of the most promising techniques. It enables an agent to determine the optimal action to take in a given state, maximizing the cumulative rewards received while interacting within a specific environment. In this framework, each satellite node acts as an agent and maintains a Q-table that stores the Q values for all possible actions and states. The Q-values are updated using the Bellman Equation. When a satellite agent receives a packet, it looks up the Q-table entry corresponding to the packet's destination node, identifies the action with the highest Q-value, and executes that action to forward the packet appropriately.

"Fast-Convergence Reinforcement Learning for Routing in LEO Satellite Networks" by Ding et al.[19] highlights that traditional routing algorithms like Dijkstra struggle to adapt in real-time to the possible dynamic nature of space networks, causing higher delays and congestion. To tackle these problems, the authors introduce a Fast-Convergence Reinforcement Learning Satellite Routing algorithm (FRL-SR). It is a procedure that employs multi-agent reinforcement learning (MARL).
In conventional Q-learning, when the state of a satellite link changes, the update is sent first to the two satellite nodes connected by the link, and then it propagates to their adjacent nodes. This process impairs the convergence of reinforcement learning algorithms, particularly in dynamic satellite networks where frequent link modifications may happen before the convergence is completed.
In FRL-SR, each satellite is an independent agent that learns to make routing decisions by observing the network states, such as link quality and queue lengths, while periodically exchanging information with neighboring satellites. The actions consist of selecting the next node from the neighboring nodes. The novelty is that the routing process involves two phases: an offline training on a static network scenario to initialize Q-tables, and then an online training to perform fine-tuning on the network feedback.
FRL-SR uses a reward that takes into account the transmission delay, the queue length, and the congestion, selecting the path with minimum delay. Simulation

results show that FRL-SR outperforms Dijkstra by reducing average latency, increasing packet delivery rates, and achieving better load distribution.

sayAn Intelligent Routing Algorithm for LEO Satellites Based on Deep Reinforcement Learning by P. Zuo et al.[29] faces the challenges of routing in dynamic Low Earth Orbit LEO satellite networks as well. The solution that is proposed is a *Deep Q-Network-based Intelligent Routing algorithm* (DQN-IR). The latter employs decentralized routing with local state information.
The DQN-IR algorithm uses Deep Reinforcement Learning to make routing decisions based on surrounding satellite conditions, such as signal to noise ratio, bandwidth, queuing delay, and distance to the destination. Each satellite acts as an agent that selects the optimal next hop using a trained DQN model, considering both the current network states and long term rewards. The reward function includes quantities such as data transmission rate, queuing delay, and hop count with the goal of reducing latency. As a result, DQN-IR demonstrates lower delay than the conventional routing algorithms. Simulations show that DQN-IR outperforms methods such as Greedy Perimeter Stateless Routing (GPSR) and Greedy Fixed Weight Routing (GFWR), achieving better delay performance by balancing short-term and long-term routing benefits. The approach has been proven to work well for LEO networks, enabling efficient local routing decisions in a dynamic environment.

Finally, *"LEO Satellite Network Routing Algorithm Based on Reinforcement Learning"* by X. Wang [30] addresses the issues presented in the previous two papers by proposing a Q-learning reinforcement learning algorithm for finding optimal transmission paths without prior knowledge of the network.
In this approach, the LEO constellation becomes a Q-learning environment. Each satellite is an agent and as usual selects the next hop based on a Q-table that is iteratively updated with feedback from the network. During training, the packets are forwarded based on Q-values, and positive feedback is propagated backward to reinforce optimal routing decisions. To enhance the algorithm's performance, two improvements are introduced:

1. the number of allowed hops is limited to fasten convergence by avoiding excessively long paths

2. to avoid the algorithm from falling into a local optimal, a dynamic greedy rate starts with exploration and then moves to exploitation.

Simulation results show that the improved Q-learning algorithm achieves faster convergence and better routing performance than traditional methods. It is successful in decreasing latency as well.

Chapter 4

Methodology and Implementation

4.1 Overview

In the following chapter, the research work is comprehensively illustrated by detailing both the design choices and the selected algorithms utilized throughout the study. To begin with, the constellation setup is thoroughly presented, along with an explanation of the Minimum Hop Count concept, which plays a crucial role in optimizing routing efficiency. Following this, a diverse collection of route path selection methods is introduced, with particular emphasis on describing several procedures employed to analyze the data and generate the outcomes. This systematic approach ensures that the entire process is reproducible, thus guaranteeing the reliability and validity of the study's findings. For further clarification, Appendix A includes the code implementations of various MATLAB functions that were useful in the analysis.

4.2 Constellation setup and Minimum Hop Count

The structure of the thesis starts from the testing of the replicability of some important results of the paper *Distributed On-Demand Routing for LEO Mega-Constellations: A Starlink Case Study* by Stock, Fraire and Hermanns [5].
First of all, the modeling of the Walker Delta constellation is outlined, defining the key parameters of the notation α: T/P/F along with other fundamental quantities such as the *argument of latitude*, also called *phase angle* and referred to as u, and the *initial longitude of the ascending node*, denoted with L_0.
The altitude of a satellite with respect to the ground is specified by the parameter h. To get the actual radius of a circular orbit, h is summed up to the semi-major axis of the Earth $r_a = 6378.137$ [km]. The latter is a constant obtained from the reference model World Geodetic System 1984 (WGS84) in which Earth is depicted as an ellipsoid [31].

All the just mentioned variables have been already described in chapter 2.4, however, it is crucial to specify that, in the case of circular orbits, both u and L_0 assume values in the interval $[-\pi, \pi[$. In particular, when u belongs to the range $[-\pi/2, \pi/2]$ it means that a satellite is *ascending*, thus moving from the south hemisphere to the north one. On the contrary, when it travels from north to south, the satellite is classified as *descending* (as illustrated in Figure 4.1).

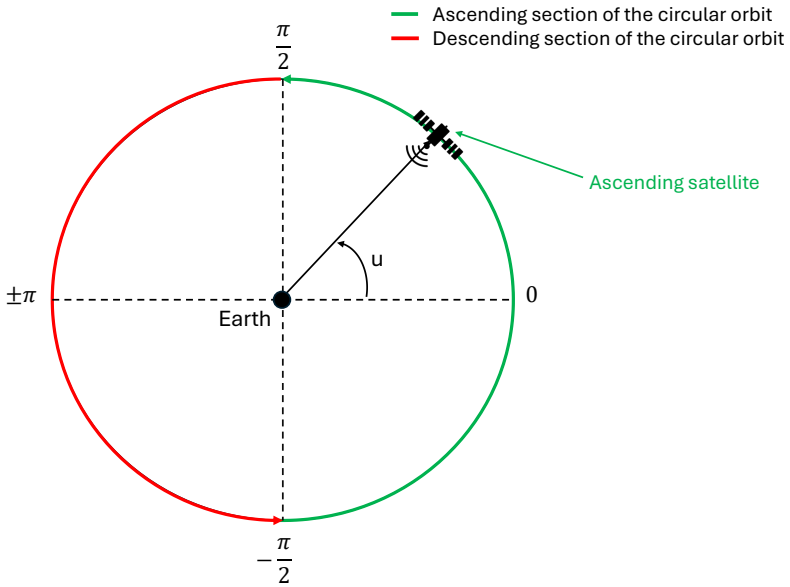


Figure 4.1: Ascending and descending sections of a circular orbit

To be consistent with paper [5], each satellite of the Walker Delta layout is uniquely identified in two ways:

- by a single index (from 1 to the total number of satellites in the constellation T),
- by the couple of indices (o, i) where o is the index (from 0 to P-1) of the orbit which the satellite under test belongs to, whereas i is the index (from 0 to Q-1) of the satellite within that orbit.

The transformations between the two ways of representing a certain satellite have been carried out with the MATLAB functions `pair_to_id` and `id_to_pair`. To convert the notation of a satellite from (o, i) to a single index, `pair_to_id` adds i to the product of o and Q (since MATLAB indices start from 1, the latter needs to be summed up as well). The `id_to_pair` function performs the reverse operation, therefore it needs as arguments both Q and the single index id, returning as outputs the couple of identifiers (o, i). To get o, the floor rounding of $(id-1)/Q$ is performed, whereas i is the remainder of the division $(id-1)/Q$.

Every satellite has 4 ISLs connecting to 4 neighbours, 2 in the same orbit and 2 in adjacent orbits. Consequently, when considering a particular satellite with indices (o, i), the previous neighbour belonging to the same orbit is $(o, (i-1) \bmod Q)$ and the successive one is $(o, (i+1) \bmod Q)$. Regarding the adjacent orbits, the satellite on the left if $o \neq 0$ is $(o-1, i)$ and $(P-1, (i-F) \bmod Q)$ otherwise. The neighbour on the right if $o \neq P-1$ is $(o+1, i)$ and $(0, (i+F) \bmod Q)$ otherwise. Functions `get_previous`, `get_successive`, `get_left`, and `get_right` aim at obtaining the just mentioned neighbours for each satellite.

There are 3 other quantities that play a significant role when delineating the geometry and the relations between satellites within the Walker Delta constellation (refer to Figure 4.2):

- The *phase difference* is the angular spacing between satellites within the same orbit. Since they are evenly distributed, it can be computed as $\Delta\Phi = \frac{2\pi}{Q} \in [0, 2\pi]$.
- The *RAAN difference* is an angular expression to define the distance between adjacent orbital planes. It follows the formula $\Delta\Omega = \frac{2\pi}{P} \in [0, 2\pi]$.
- The *phase offset* $\Delta f = \frac{2\pi F}{PQ} \in [0, 2\pi[$ that determines the "gain" in latitude of a satellite with respect to another on an adjacent plane (therefore with respect to the left and right neighbours) and not in the same orbit. When $\Delta f = 0$ it means they are aligned (meaning they have the same latitude). Moreover, the

constellation is symmetric, hence each satellite has the same offset Δf from the corresponding ones belonging to adjacent planes and $P \cdot \Delta f$ is a multiple of $\Delta\Phi$.

Thanks to the definition of the *phase difference*, the *RAAN difference* and the *phase offset*, it is possible to write both the phase angle and the initial longitude of the ascending node of a generic satellite as functions of the indices pair (o, i):

- $u = f(o \cdot \Delta\Omega + i \cdot \Delta\Phi)$
- $L_0 = f(o \cdot \Delta\Omega)$

where the function $f(x) = ((x + \pi) \bmod 2\pi) - \pi$ is just a type of normalization, so that the result always belongs to the interval $[-\pi, \pi[$
In MATLAB each constellation configuration is implemented thanks to the built-in functions `satelliteScenario` and `walkerDelta` which require α, T, P, F, the radius from the center of the Earth, and the initial longitude of the ascending node equal to zero.

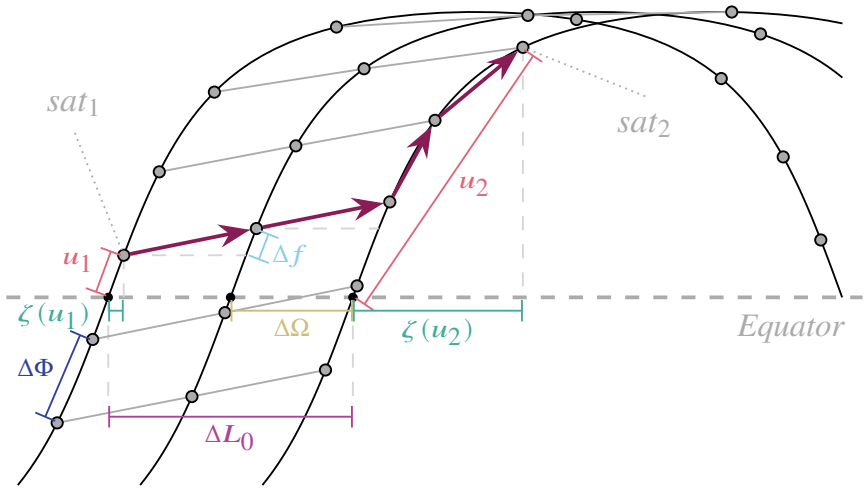


Figure 4.2: Section of a Walker Delta constellation with modelling parameters [5]

The focus of the paper by Stock et al. [5] is the use of the concept of the Minimum Hop Count introduced by Chen et al.[17] to implement a new heuristic called *DisCoRoute* that handles routing for Walker Delta constellations launched by Starlink.
The above-mentioned parameters and relations are crucial in the computation of the Minimum Hop Count. Given a source satellite belonging to a Walker Delta

pattern, the Minimum Hop Count is the minimum number of hops that the routing path must cross to reach a designated destination satellite. The hops are either horizontal when the link between two satellites is between two adjacent planes, also known as *inter-link* (therefore obtained with `get_left` and `get_right` functions) or vertical when the link is within the same orbit, the *intra-link* (thus provided by `get_previous`, `get_successive`).

The demonstration below is extracted from [5].
Assuming that:

- $\Delta L_0 = (L_{0,dest} - L_{0,src}) \bmod 2\pi$ is the total angular difference of the initial longitude of the ascending node of the source and the destination following the East direction ($2\pi - \Delta\Omega$ selecting West direction),
- $\Delta\Omega$ is the distance between adjacent orbital planes.

Then, the number of horizontal hops is given by the ratios:

$$H_{h,E} = \left\lfloor \frac{\Delta L_0}{\Delta\Omega} \right\rceil \tag{4.1}$$

$$H_{h,W} = \left\lfloor \frac{2\pi - \Delta L_0}{\Delta\Omega} \right\rceil \tag{4.2}$$

where and $\lfloor\cdot\rceil$ is a notation indicating the so-called *commercial rounding* that performs $\lfloor x \rceil = sgn(x)\,\lfloor |x| + 0.5 \rfloor$.

Suppose now that a satellite under test with phase angle u_1 has to be connected with another one that is its closest neighbour in the North-East direction. The latter will have a phase angle expressed by:

$$u_2 = u_1 + (H_E \cdot \Delta f) + (H_{NE} \cdot \Delta\Phi) \tag{4.3}$$

The term $(H_{NE} \cdot \Delta\Phi)$ is denominated Δu_E and corresponds to:

$$\Delta u_E = (u_2 - u_1 - (H_E \cdot \Delta f)) \bmod 2\pi \tag{4.4}$$

Obviously, in case of direction North-West:

$$\Delta u_W = (u_2 - u_1 - (H_W \cdot \Delta f)) \bmod 2\pi \tag{4.5}$$

Finally, given Δu_E and Δu_W it is possible to derive the vertical hops and the directions with the equations:

$$H_{v,NE} = \left\lfloor \frac{\Delta u_E}{\Delta\Phi} \right\rceil \tag{4.6}$$

$$H_{v,NW} = \left\lfloor \frac{\Delta u_W}{\Delta \Phi} \right\rceil \tag{4.7}$$

$$H_{v,SE} = \left\lfloor \frac{2\pi - \Delta u_E}{\Delta \Phi} \right\rceil \tag{4.8}$$

$$H_{v,SW} = \left\lfloor \frac{2\pi - \Delta u_W}{\Delta \Phi} \right\rceil \tag{4.9}$$

To determine the Minimum Hop Count and identify the corresponding directions to follow, the smallest value among four different combinations of horizontal and vertical hops sums should be selected:

$$\min \{H_{v,NE} + H_{h,E}, H_{v,NW} + H_{h,W}, H_{v,SE} + H_{h,E}, H_{v,SW} + H_{h,W}\} \tag{4.10}$$

For completeness, in Figure 4.2, the function $\zeta(u)$ is also present, where $\zeta(u) = \arctan(\cos(\alpha) \cdot \tan(u))$, with 0 for the ascending segment and π for the descending segment. $\zeta(u)$ is the angular difference in longitude of a satellite with respect to the ascending node of its orbit.

4.3 Route Path Selection Methods

Let's assume that a Walker Delta constellation has been generated with a precise choice of parameters (α, T, P, F, h). Then, it is plotted both in 3D with MATLAB built-in function `satelliteScenarioViewer` and in a 2D grid with an ad hoc (for that purpose only) function.

This section provides an in-depth exploration of the routing methodologies and techniques employed to establish a connection between two distinct satellites within the aforementioned constellation. Each routing strategy is meticulously examined, with a step-by-step explanation to ensure a clear understanding of its underlying mechanisms.
In general, all the procedures depicted in this chapter are implemented such that they take as inputs the source and destination satellites that have to be connected and return as outputs their algorithm's name and the routing paths both as pair of indices (o, i) and as the single index notation. The route path first entry is the source node (so it has the size of the sum of vertical and horizontal hops plus 1).

4.3.1 Trivial Routing algorithm

This method takes as inputs, besides the already mentioned ones, the results of the Minimum Hop Count function (therefore the minimum sum of the combinations of horizontal and vertical hops as well as their selected directions) and some

parameters of the constellation (P, Q, F). Next, it computes a simple ("trivial") routing path across the constellation based on the Minimum Hop Count. Starting from the source, it moves based on the specified vertical and horizontal hop counts. The algorithm handles all the horizontal hops first, whether they are either 'W' (West) or 'E' (East) according to the chosen direction, and then proceeds with the remaining all vertical ones, moving 'N' (North) or 'S' (South). P, Q and F are used to tackle the wrap-around of the constellation structure.

The procedure accepts *first_hops* as an extra argument. It is used to extend the flexibility of Trivial Routing algorithm. This modification supports the development of a new heuristic designed to address the behaviour of routing paths near Arctic and Antarctic regions. Essentially, it allows a choice between prioritizing either all the vertical hops or horizontal ones initially, followed by all the others.

4.3.2 Flip-Coin Routing algorithm

This strategy follows the same principles outlined in the CoinFlip algorithm described in [5]. It takes the same inputs of Trivial Routing algorithm and it also utilizes the Minimum Hop Count. By determining the total number of horizontal and vertical hops, along with the respective directions, the algorithm effectively connects the source and destination satellites. The outputs consist of routes represented by pairs of indices (o, i), as well as single satellite node indices. The core decision-making process, thus how the destination is reached, relies on coin-flipping, where the choice between moving horizontally or vertically is made with equal probability at each satellite node, introducing an element of randomness to the path selection.
A vertical token and a horizontal token are set with the corresponding total number of vertical and horizontal hops obtained by the Minimum Hop Count, then the algorithm loops and at each iteration checks if there are still available tokens. For instance, if there are no more vertical hops, the choice is forced on the horizontal ones.

As an enhancement to this strategy, the choice at each satellite node is no longer purely random between the two directions, but instead weighted based on the remaining hops that still have to be traversed. This approach offers greater flexibility compared to the equal-probability coin flipping. Since the Minimum Hop Count constrains the search space to a rectangular grid where hops follow a Manhattan distance fashion between the source and the destination which are 2 vertices, the methodology is more aligned with the routing context. For example, if 99 vertical hops and only 1 horizontal hop remain, the algorithm will choose the vertical direction 99% of the times and the horizontal direction just 1%. This method reduces unnecessary detours and adapts dynamically to the situation,

making it a more efficient alternative to the random selection approach.

4.3.3 DisCoRoute Routing algorithm

This technique is the core of the paper *Distributed On-Demand Routing for LEO Mega-Constellations: A Starlink Case Study* by Stock, Fraire and Hermanns [5]. It is based on the design of the constellation, exploiting the fact that the *intra-links* (within the same orbit) are constant in magnitude whereas *inter-links* (between adjacent orbital planes) decrease the closer they are to the polar regions. Furthermore, the conventional Dijkstra's solution does not account for the Minimum Hop Count. As a result, when considering the total end-to-end delay rather than just the propagation delay, having even one hop more may increase significantly the latency (as it is shown in the following chapter).

It takes as inputs the source and destination satellite nodes, some parameters of the constellation (P, Q, F), the Minimum Hop Count as well as the directions of the hops, and the latitude of each satellite. The output is a sub-optimal route between two satellite nodes, constructed with horizontal (H_h) and vertical (H_v) hops and expressed both as satellite pairs and single satellite IDs.
Firstly, the algorithm first initializes the indices pairs of the source and destination nodes (o_src, i_src, o_dst, i_dst), as well as 2 sections of the final route, called `route_s` and `rout_t`, starting respectively one from the source and one from the destination.
Then, it verifies if both nodes are either *ascending* or *descending*, using the function `isAscending`. The check is pivotal because it splits the procedure into 2 sections:

- the routing case is A2A (*ascending* to *ascending*), described in Algorithm 1
- the routing case is A2D (*ascending* to *descending*), shown in Algorithm 2.

The considerations that are done are equivalent for the scenarios D2D (*descending* to *descending*) and D2A (*descending* to *ascending*).

In the A2A case, the algorithm manages horizontal hops by iteratively calculating rewards based on the latitudes of neighboring nodes. At each step, the route construction alternates between the source and destination, depending on which path provides a lower reward (determined by latitude differences). Once the horizontal hops are completed, 2 route sections are generated: the final satellites of each segment are positioned within the same orbit, and if necessary, vertical hops are appended on that plane.

For the A2D case, the method reverses the logic, first performing vertical hops. Similarly to the A2A case, it computes the latitude-based rewards for both the

source and destination paths, building the vertical route segments iteratively. After completing all the vertical hops, horizontal hops are added to connect the nodes on the same orbital plane.

Algorithm 1 Disco Routing Algorithm (A2A Case)

Require: latitudes, src_node, dst_node, MinHopCount, P, Q, F
Ensure: RoutePath, route_singleid_path, RoutingName

Extract o_{src}, i_{src} from src_node
Extract o_{dst}, i_{dst} from dst_node
Set horizontal (H_h) and vertical (H_v) hops from MinHopCount
Initialize $route_s \leftarrow src\_node$, $route_t \leftarrow dst\_node$
if isAscending($o_{src}, i_{src}, PQ, P, F$) == isAscending($o_{dst}, i_{dst}, PQ, P, F$) **then** ▷ A2A Case: Horizontal Routing
    Set $i \leftarrow 0$, $j \leftarrow H_h$
    for $h = 1$ to H_h **do**
        Compute $latid_{i,0}$ and $latid_{i+1,0}$ for $route_s$
        Compute $latid_{j,H_v}$ and $latid_{j-1,H_v}$ for $route_t$
        Calculate reward_s and reward_t based on latitude values
        if reward_s < reward_t **then**
            Add a hop to $route_t$ (left or right based on direction)
            Update $j \leftarrow j - 1$
        else
            Add a hop to $route_s$ (left or right based on direction)
            Update $i \leftarrow i + 1$
        end if
    end for
    Ensure $i = j$ (i.e., same orbital plane is reached)
    if $H_v > 0$ **then**
        Add vertical hops to $route_s$
        for $v = 1$ to $H_v - 1$ **do**
            Add vertical hops based on North/South direction
        end for
    end if
    Concatenate $route_s$ and reversed $route_t$ to form $RoutePath$
    Convert $RoutePath$ coordinates to MATLAB indexes using $pair\_to\_id$
    Set $route\_singleid\_path \leftarrow RoutePath$ converted to single IDs
end if
Set $RoutingName \leftarrow$ "with DiscoRouting" **return** $RoutePath$, $route\_singleid\_path$, $RoutingName$

Algorithm 2 Disco Routing Algorithm (A2D Case)

Require: latitudes, src_node, dst_node, MinHopCount, P, Q, F
Ensure: RoutePath, route_singleid_path, RoutingName

Extract o_{src}, i_{src} from src_node
Extract o_{dst}, i_{dst} from dst_node
Set horizontal (H_h) and vertical (H_v) hops from MinHopCount
Initialize $route_s \leftarrow src\_node$, $route_t \leftarrow dst\_node$
if isAscending$(o_{src}, i_{src}, PQ, P, F) \neq$ isAscending$(o_{dst}, i_{dst}, PQ, P, F)$ **then** ▷ A2D Case: Vertical Routing
    Set $i \leftarrow 0$, $j \leftarrow H_v$
    for $h = 1$ to H_v **do**
        Compute $latid_{0,i}$ and $latid_{0,i+1}$ for $route_s$
        Compute $latid_{H_h,j}$ and $latid_{H_h,j-1}$ for $route_t$
        Calculate reward_s and reward_t based on latitude values
        if reward_s $<$ reward_t **then**
            Add a vertical hop to $route_s$ based on direction
            Update $i \leftarrow i + 1$
        else
            Add a vertical hop to $route_t$ based on direction
            Update $j \leftarrow j - 1$
        end if
    end for
    if $H_h > 0$ **then**
        Add horizontal hops to $route_s$
        for $horiz\_hop = 1$ to $H_h - 1$ **do**
            Add horizontal hops based on East/West direction
        end for
    end if
    Concatenate $route_s$ and reversed $route_t$ to form $RoutePath$
    Convert $RoutePath$ coordinates to MATLAB indexes using $pair\_to\_id$
    Set $route\_singleid\_path \leftarrow RoutePath$ converted to single IDs
end if
Set $RoutingName \leftarrow$ "with DiscoRouting" **return** $RoutePath$, $route\_singleid\_path$, $RoutingName$

4.3.4 Dijkstra's Routing algorithm

Dijkstra's Routing algorithm [18] applied to satellite networks finds the shortest path with minimum propagation delay from a chosen source satellite to its destination. The satellite constellation can be considered as a weighted graph where the weights associated with the links between satellites are never negative. Therefore, Dijkstra's solution is optimal, meaning that it grants the shortest path through an exhaustive exploration of all the reachable neighbour satellites of the node under test, marking them as "visited" only if they have the minimum cumulative distance from the source. It also keeps track of the previous crossed nodes to enable path reconstruction.

In essence, the function has the positions of satellites, the total number of satellites, the source and destination nodes and some parameters of the constellation as inputs. As usual, it returns the route path and the algorithm name. There are three arrays that have the size of the total number of satellites. They are initialized, `distance` is set to infinite, `prev` to *Not-a-Number* and `visited` to *False*, with the distance of the source set to zero as a starting point. Then, the min-heap priority queue `Q_dijkstra` is generated, keeping track of nodes and distances.
A while loop iterates over the unvisited nodes getting the one with minimum distance and setting it as "visited". The condition to leave the loop is when the node is the destination satellite. The neighbours of the current satellite are analyzed by computing the Euclidean distance and if a minimum distance is found, then the distance vector, the array of predecessors and the priority queue are updated. Since the chain of previous nodes has been recorded, it is possible to reconstruct the path starting from the destination up to the source.
For further explanation, the pseudo-code is shown in Algorithm 3.

Unlike the algorithms that rely on the minimum hop count computation, Dijkstra's procedure calculates the route which is the shortest in terms of overall distance and propagation delay, however it may happen that it does not traverse the minimum number of nodes. The implications of this fact require some further analysis and observations, also in terms of overall performance, that are widely discussed in the "Results and Discussion" section (Chapter 5).

4.3.5 DijkstraHop Routing Algorithm

This function is implemented in a similar way with respect to the traditional Dijkstra's algorithm, yet with a crucial distinction: rather than being optimized for finding the shortest path in terms of distance, it focuses specifically on minimizing the number of hops. The core modification lies in how neighboring nodes are examined. Unlike the standard approach, where edge weights may vary and are

Algorithm 3 Dijkstra's Algorithm for Satellite Routing

Input: *positions*, *num_satellites*, *src_node*, *dst_node*, P, Q, F
Output: *RoutePath*, *route_singleid_path*, *RoutingName*
$src\_index \leftarrow$ pair_to_id(Q, *src_node*)
$dst\_index \leftarrow$ pair_to_id(Q, *dst_node*)
Initialize $distance \leftarrow \infty$, $prev \leftarrow NaN$, $visited \leftarrow$ False for all satellites
$distance(src\_index) \leftarrow 0$
Initialize priority queue $Q\_dijkstra \leftarrow [(1 : num\_satellites)', distance]$
while any satellite is not visited **do**
    Extract node u with minimum distance from $Q\_dijkstra$
    Remove u from $Q\_dijkstra$
    Mark u as visited
    if $u == dst\_index$ **then**
        break
    end if
    Get (o, i) pair from u using id_to_pair(Q, u)
    Get neighbours of u from get_4neighbours(P, Q, F, o, i)
    for each neighbour v of u **do**
        if v is not visited **then**
            Calculate Euclidean distance $alt \leftarrow distance(u) + \text{norm}(positions(:, u) - positions(:, v))$
            if $alt < distance(v)$ **then**
                Update $distance(v) \leftarrow alt$
                Update $prev(v) \leftarrow u$
                Update priority queue $Q\_dijkstra(v) \leftarrow alt$
            end if
        end if
    end for
end while
Initialize empty *route_singleid_path*
Reconstruct the shortest path starting from *dst_index*:
while u is not NaN **do**
    Prepend u to *route_singleid_path*
    $u \leftarrow prev(u)$
end while
Initialize *RoutePath* $\leftarrow$ empty array
for each index in *route_singleid_path* **do**
    Convert index to (o, i) pair and append to *RoutePath*
end for
RoutingName $\leftarrow$ ' with Dijkstra'
return *RoutePath*, *route_singleid_path*, *RoutingName*

cumulatively summed to determine the shortest distance, here, every link is treated as having a unitary weight.

To put this into context, the change in the pseudo-code is relatively localized, affecting primarily the section responsible for exploring neighboring nodes and calculating the cumulative cost of the path. Here, the weight of each edge is treated as equal, reducing the complexity of edge comparison to merely counting steps (Algorithm 4). In essence, the algorithm navigates the graph by assigning each edge a uniform weight of one, simplifying the computation and optimizing for path efficiency in terms of the number of intermediate nodes encountered. For a more comprehensive evaluation of its performance, detailed results and comparative metrics will be discussed in the subsequent chapter.

Algorithm 4 Neighbours exploration for DijkstraHop

Let h be the current number of hops from the source to node u
Let $numOfHops(v)$ be the minimum number of hops from the source to node v
for each neighbour v of u **do**
  if v is not visited **then**

    $alt \leftarrow h + 1$ ▷ Each edge represents one hop
    if $alt < numOfHops(v)$ **then**
      Update $numOfHops(v) \leftarrow alt$
      Update $prev(v) \leftarrow u$ ▷ Set the predecessor of v to u
      Update priority queue $Q\_dijkstra(v) \leftarrow alt$
    end if
  end if
end for

4.3.6 DijkstraTrans Routing algorithm

This method represents another variation of the classical Dijkstra's algorithm, originally designed to optimize the shortest path in terms of physical distance. However, in this implementation, the focus shifts towards minimizing the overall end-to-end delay. In the framework of this investigation, the end-to-end delay is the sum of two key components: the traditional propagation delay and the transmission time given by each node along the path.
The propagation delay is preserved as it is in the original Dijkstra's algorithm, reflecting the time taken for signals to travel across distances between satellites. However, this implementation introduces an additional term: the transmission time. This value, though fixed and externally configurable, plays a critical role

in capturing the full scope of delays experienced during communication between satellites.
In this modified algorithm, the transmission time is incorporated directly into the core path-finding logic. In particular, within the for-loop section where the neighboring satellites of a node under investigation are considered, the transmission time is treated as an additional distance factor. To maintain consistency with the propagation delay metric, the transmission time is multiplied by the speed of light, effectively converting it into a distance. By summing these two components, the propagation delay and the transmission time, the algorithm is capable of selecting paths that offer the best performance in terms of overall end-to-end delay. For a more detailed understanding of the implementation, the pseudo-code (Algorithm 5) shows the modifications that are applied in the loop responsible for evaluating neighboring satellites. In Chapter 5 the algorithm performance and evaluation of the results will be shown as well.

Algorithm 5 Neighbours exploration for DijkstraTrans

```
for each neighbour v of u do
    if v is not visited then
        Compute the distance between u and v, plus transmission time
        alt ← d + norm(positions(:, u) − positions(:, v)) + transTime × c
        if alt < distance(v) then
            distance(v) ← alt              ▷ Update the shortest distance to node v
            Update prev(v) ← u             ▷ Set the predecessor of v to u
            Q_dijkstra(v) ← alt            ▷ Update priority queue
        end if
    end if
end for
```

4.3.7 New heuristic: T.A.A.T. enhancement

Algorithm T.A.A.T. stands for *Targeted for Arctic and Antarctic Tracking*. It adapts the path calculation based on the latitude of the source and destination satellites, potentially leveraging polar satellites when the source and destination are near the poles. This technique is modeled to handle routing between satellites in the polar regions of a constellation, especially when both the source and the destination satellites are situated near the poles. The goal of the algorithm is to efficiently use the satellites in the Arctic and Antarctic areas to minimize hops and improve routing performance.

Initially, the algorithm identifies the source and destination satellites' latitude. Subsequently, it checks two thresholds to trigger the algorithm:

- Is the latitude of both satellites greater than 62.5°? If yes, it means they are close to a polar region. Noteworthy is the fact that the value of the threshold is positive for the north pole and negative for the south pole.
- Are there enough hops between the source and destination satellites?

If these conditions are met, the T.A.A.T. algorithm is executed; otherwise, it defaults to the DisCoRoute algorithm.
T.A.A.T. first identifies satellites located near the poles (above 82° latitude) and calculates the Minimum Hop Count from the source satellite to these polar satellites. Next, it calculates the minimum hops from the destination satellite to the same set of polar satellites. By comparing the hop counts, the algorithm selects the best polar satellite pair (one close to the source and one close to the destination) that minimizes the total number of hops.

The routing process is divided into three segments:

- From the source satellite to the selected source polar satellite.
- From the source polar satellite to the destination polar satellite.
- From the destination polar satellite to the final destination satellite.

The section from the source satellite to the polar source is calculated by computing the Minimum Hop Count from the source to a collection of polar satellites, then selecting the one that is closest (in terms of hops) and finally computing the route with Trivial algorithm. The same process is followed for the path from the polar destination to the actual destination satellite.
For the polar-to-polar segment, T.A.A.T. selects routing paths through the poles by minimizing a custom metric that incorporates the latitudes of the polar satellites and the number of vertical hops. This ensures that the routing path takes full advantage of the polar constellation's geometry. Finally, the algorithm concatenates the three segments into a single route, offering a final path through the polar region.
If the polar conditions are not met, the default DisCoRoute algorithm is used.
To improve the readability of the overall Algorithm, the latter has been split into 2 segments:

- the first section handles the input processing, if conditions and the basic structure (Part 1 - Algorithm 6)
- the second one focuses on the routing in the polar regions and constructing the final route (Part 2 - Algorithm 7)

Algorithm 6 TAAT Routing Algorithm (Part 1)

Input: latitudes, src_node, dst_node, P, Q, F
Output: RoutePath, route_singleid_path, RoutingName, used_taat
PQ ← P * Q ▷ Total number of satellites
Extract source and destination satellite IDs
$o_{\text{src}}, i_{\text{src}}$ ← src_node
$o_{\text{dst}}, i_{\text{dst}}$ ← dst_node
src_id ← pair_to_id(Q, o_{src}, i_{src})
dst_id ← pair_to_id(Q, o_{dst}, i_{dst})
src_lat ← latitudes[src_id]
dst_lat ← latitudes[dst_id]
Compute Minimum Hop Count between Source and Destination
MinHopCount ← min_hop_count_v3(PQ, P, F, o_{src}, i_{src}, o_{dst}, i_{dst})
Check Conditions for T.A.A.T. Routing
lat_threshold ← 62.5
lat_condition_north ← (src_lat > lat_threshold) **and** (dst_lat > lat_threshold)
lat_condition_south ← (src_lat < -lat_threshold) **and** (dst_lat < -lat_threshold)
lat_condition ← lat_condition_north **or** lat_condition_south
vertical_hop_threshold ← 5
hop_condition ← MinHopCount.H_h_H_v[2] > vertical_hop_threshold
taat_condition ← lat_condition **and** hop_condition
if taat_condition **then**
    used_taat ← 1
    Identify Pole Satellites
    if lat_condition_north **then**
        pole_satellites ← find(latitudes > 82)
    else if lat_condition_south **then**
        pole_satellites ← find(latitudes < -82)
    end if
else
    used_taat ← 0
    Apply DiscoRouting
    RoutePath ← runDiscoRouting(latitudes, src_node, dst_node, MinHopCount, P, Q, F)
    Go to End
end if

Algorithm 7 TAAT Routing Algorithm (Part 2)

Step 1: Compute hops from source to all pole satellites
for all ps in pole_satellites **do**
    $o_{\text{pole}}, i_{\text{pole}} \leftarrow$ id_to_pair(Q, ps)
    MinHopCountPoles $\leftarrow$ min_hop_count(PQ, P, F, $o_{\text{src}}, i_{\text{src}}, o_{\text{pole}}, i_{\text{pole}}$)
    num_hop_src_to_pole_satellites[ps] $\leftarrow$ sum(MinHopCountPoles.H_h_H_v)
end for
Step 2: Compute hops from destination to all pole satellites
for all ps in pole_satellites **do**
    $o_{\text{pole}}, i_{\text{pole}} \leftarrow$ id_to_pair(Q, ps)
    MinHopCountPoles $\leftarrow$ min_hop_count(PQ, P, F, $o_{\text{pole}}, i_{\text{pole}}, o_{\text{dst}}, i_{\text{dst}}$)
    num_hop_dst_to_pole_satellites[ps] $\leftarrow$ sum(MinHopCountPoles.H_h_H_v)
end for
Step 3: Select the best 5 pole satellites for source and destination
closest_src_pole $\leftarrow$ mink(num_hop_src_to_pole_satellites, 5)
closest_dst_pole $\leftarrow$ mink(num_hop_dst_to_pole_satellites, 5)
Step 4: Evaluate best pole satellite pair (source and destination)
Initialize num_hops_src_to_dst_with_poles[5][5] $\leftarrow$ 0
for all src_pole in closest_src_pole **do**
    for all dst_pole in closest_dst_pole **do**
        Compute hops for source $\rightarrow$ src_pole $\rightarrow$ dst_pole $\rightarrow$ destination
        Store result in num_hops_src_to_dst_with_poles
    end for
end for
Select src_pole_best and dst_pole_best that minimize num_hops_src_to_dst_with_poles
Construct the final route
RoutePath $\leftarrow$ source $\rightarrow$ src_pole_best $\rightarrow$ dst_pole_best $\rightarrow$ destination
Convert route to single IDs
for all i in RoutePath **do**
    route_singleid_path[i] $\leftarrow$ pair_to_id(Q, RoutePath[i])
end for
RoutingName $\leftarrow$ ' with T.A.A.T.'
Return: RoutePath, route_singleid_path, RoutingName, used_taat

Chapter 5

Results and Discussion

5.1 Overview

The entire project has been developed on MATLAB Release 2023b. All the constellation patterns and the tested algorithms have been implemented with MATLAB programming language. Their performance and the selected benchmarks have been assessed through multiple simulations. The utilized computing machine has a Windows 11 OS with CPU 11th Gen Intel Core i7 at 2.80GHz, and 16 GB of RAM.

In this chapter several constellation configurations are selected and illustrated by plotting them in 3D with MATLAB function `satelliteScenarioViewer`. The algorithms that have been outlined in Chapter 4, *Methodology and Implementation*, are plotted in 3D and 2D. They have been evaluated and compared taking the conventional Dijkstra's solution as a baseline. Firstly, the goal is to test the reproducibility of the outcomes obtained in [5] (in terms of end-to-end delay and computational time) of various optimal and sub-optimal routing algorithms on the constellation. Then, a performance analysis of the procedures in terms of distinct constellation conditions (different inclinations, altitudes and topologies) is shown. Finally, the description of a new heuristic that handles routing in the polar regions

is provided.

The constellation designs examined in this thesis are:

- LEO Walker Delta configuration 53°:1584/72/39 at 550 km from the FCC Filing SAT-MOD-20190830 document sent by Space Exploration Holding LLC [32]
- VLEO Walker Delta pattern 96.9°:2000/40/21 at 360 km from from the FCC Filing SAT-AMD-20210818 document sent by Space Exploration Holding LLC [33]

The phasing factor F is calculated by finding the value of the *phase offset* from the obtained documentation (both [32] and [33] have a technical parameters attachment that can be opened with *Microsoft Access*) and then inverting the formula $\Delta f = \frac{2\pi F}{PQ} \in [0, 2\pi[$.

Note that as a further constellation, also LEO 60°:500/25/5 at 550 km has been considered for plotting purposes.

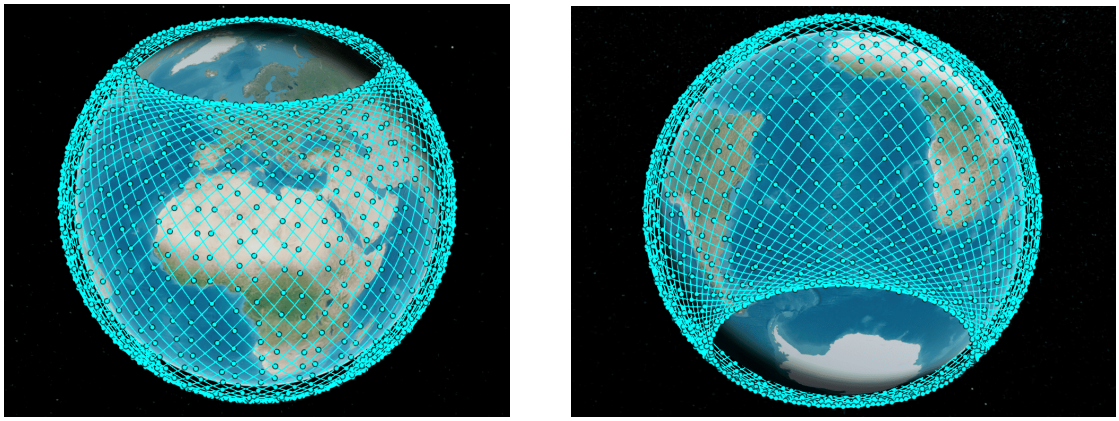
5.2 Constellation 3D plots

In this project, two main Starlink constellation patterns are examined. They are depicted in Figures 5.1 and 5.2. As elaborated in Chapter 2, these constellations are arranged using the Walker Delta configuration, characterized by a distinctive flower-like distribution of orbital planes that ensures near-global coverage. Indeed, this type of design provides effective communication service over the Earth's surface, with the exception of the polar regions. The extent of these uncovered polar gaps is influenced primarily by the inclination angle chosen for the constellation (the higher the absolute value of α, the smaller the uncovered gap).

The LEO arranged setup is positioned at an altitude of 550 km, while the VLEO configuration operates at 360 km. This altitude difference highlights the trade-offs between coverage area, satellite count, orbital mechanics, and operational complexity that must be carefully balanced in the design of satellite constellations. Lower altitudes, as in the case of VLEO, offer reduced latency and improved resolution for communication systems, but at the cost of requiring more satellites and increased operational demands due to faster orbital dynamics.

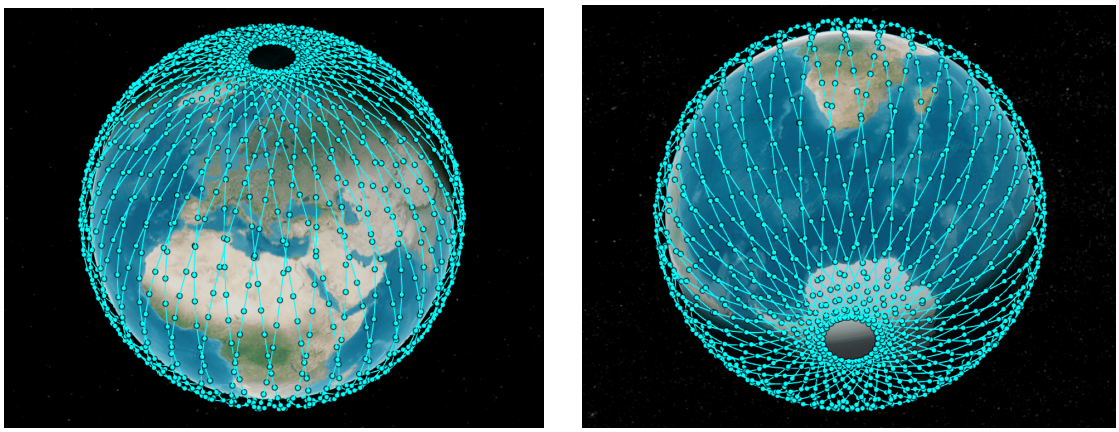
As already mentioned, another critical consideration for VLEO constellations is the stronger gravitational force experienced at lower altitudes. As a matter of fact, satellites in VLEO must contend with a stronger gravitational attraction, consequently, they orbit around the Earth at a faster rate, resulting in shorter orbital

periods compared to satellites in LEO. This increased velocity also contributes to faster orbital decay, causing shorter satellite lifespans.



(a) (1) **(b)** (2)

Figure 5.1: LEO Walker Delta constellation 53°:1584/72/39 at 550 km



(a) (1) **(b)** (2)

Figure 5.2: VLEO Walker Delta constellation 96.9°:2000/40/21 at 360 km

When visualizing the routing paths in the 3D representations, the source and destination satellites are marked with red dots, while the satellites traversed along the path are shown in green.
The connections between them are illustrated using the function `plot_path` (whose full implementation is available in Appendix A).
Links between satellites in adjacent planes, known as *interplane links*, correspond to horizontal hops (H_h) and are shown in green.

Links between satellites within the same orbit, referred to as *intraplane links*, represent vertical hops (H_v) and are shown in red.

Figures 5.3 and 5.4 refer to `Trivial` Routing and `FlipCoin` Routing applied from source satellite (21,10) to destination satellite (23,13). Both algorithms use the Minimum Hop Count as input, which is set to 5, consisting of 2 horizontal hops and 3 vertical hops, with the selected direction being North-East.
Although both strategies have the same Minimum Hop Count and direction, they take different paths, as illustrated in Tables 5.1 and 5.2 with the notation (o, i) in the first two columns and the single satellite IDs in the remaining one. The `Trivial` algorithm completes all the horizontal hops first before moving on to the vertical hops. In contrast, the `FlipCoin` algorithm selects its path based on coin flips, giving equal probability to each direction.

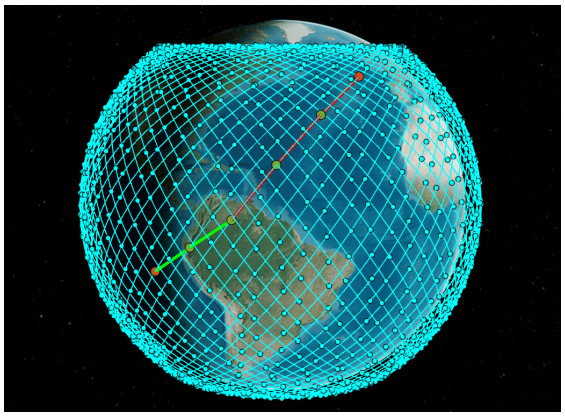


Figure 5.3: Trivial Route Path on LEO 53°:1584/72/39 from (21,10) to (23,13)

o	i	Single ID
21	10	473
22	10	495
23	10	517
23	11	518
23	12	519
23	13	520

Table 5.1: Trivial Route Path on LEO from (21,10) to (23,13)

Figure 5.5 is the result of the traditional `Dijkstra` Routing algorithm applied to find the shortest path in terms of overall distance (thus, minimum propagation delay) from satellite (21,10) to (23,13). The route with the satellite nodes represented in both (o,i) and single IDs notations is in table 5.3.

Noteworthy is the fact that `DisCoRoute` converges to the same result using both Minimum Hop Count (obtaining 3 horizontal hops and 2 vertical ones, with direction North-East) and the part of the strategy that handles A2A (Ascending to Ascending) satellites, constructing two routes of horizontal hops, one from the source and one from the destination, and finally connecting them with a route of vertical links (if necessary).

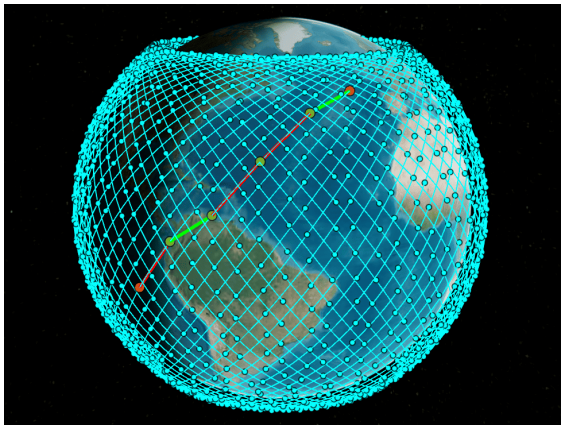


Figure 5.4: Flip Coin Route on LEO 53°:1584/72/39 from (21,10) to (23,13)

o	i	Single ID
21	10	473
21	11	474
22	11	496
22	12	497
22	13	498
23	13	520

Table 5.2: Flip Coin Route Path on LEO from (21,10) to (23,13)

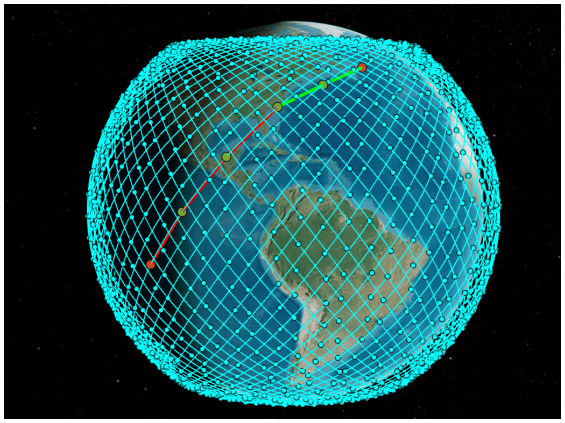


Figure 5.5: Dijkstra Route Path on LEO 53°:1584/72/39 from (21,10) to (23,13)

o	i	Single ID
21	10	473
21	11	474
21	12	475
21	13	476
22	13	498
23	13	520

Table 5.3: Dijkstra Route Path on LEO from (21,10) to (23,13)

Figure 5.6 shows `DisCoRoute` applied in the A2D (Ascending to Descending) case from satellite (24,12) to (26,16). The Minimum Hop Count is 6 with 4 vertical hops and 2 horizontal ones, direction North-East. `DisCoRoute` in the A2D (or D2A) case exploits the conjecture that the closer the links to the poles, the shorter the horizontal hops [5]. Therefore, firstly it selects the vertical hops in order to reach the poles starting both from the source and the destination, then connects the resulting two routes (one from the source and one from the destination) with one made of only horizontal hops.

To provide further clarity, Figures 5.7a and 5.7b show two distinct scenarios where the `DisCoRoute` algorithm selects paths based exclusively on vertical hops or

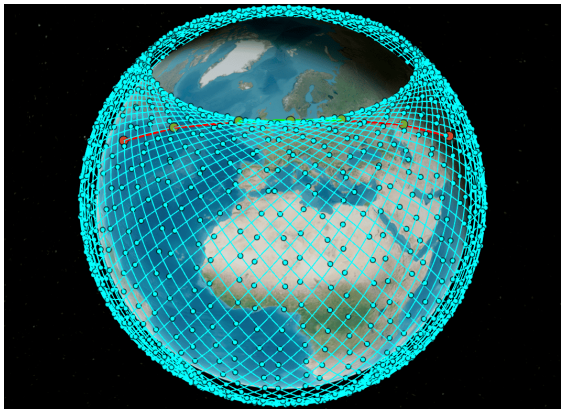
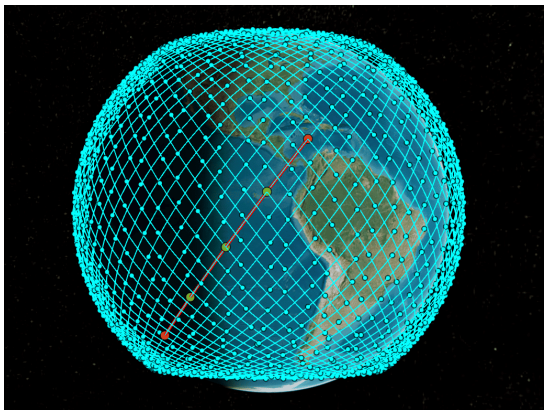


Figure 5.6: DisCoRoute Route Path on LEO 53°:1584/72/39 from (24,12) to (26,16)

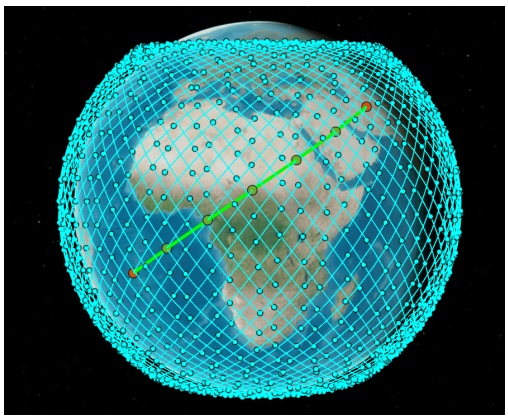
o	i	Single ID
24	12	541
24	13	542
24	14	543
25	14	565
26	14	587
26	15	588
26	16	589

Table 5.4: DisCoRoute Route Path on LEO from (24,12) to (26,16)

horizontal hops. In the first scenario, depicted in Figure 5.7a, the algorithm chooses a path that involves only vertical hops. This path connects the starting node at (24,6) to the destination node at (24,10), maintaining the same orbit index while moving vertically across the grid. In contrast, Figure 5.7b depicts a case where the algorithm opts for a path composed of horizontal hops. Here, the movement is from node (47,21) to node (41,21), where the satellite index within an orbit remains constant, and the algorithm traverses only horizontally. These two figures provide clear visual examples of how Minimum Hop Count-based algorithms can adapt the routing strategy based on the specific network scenario.



(a) DisCoRoute Route Path on LEO from (24,6) to (24,10) only vertical hops



(b) DisCoRoute from (47,21) to (41,21) only horizontal hops

Figure 5.7: DisCoRoute Route Path on LEO 53°:1584/72/39 at 550 km

The same considerations can be applied to the Walker Delta configuration VLEO 96.9°:2000/40/21 at 360 km.
In Figure 5.8 `Trivial` Routing is applied to reach destination satellite (0,18) from source satellite (37,5). The algorithm takes as input the Minimum Hop Count, therefore the route will cross a total of 11 hops, with direction South-East (3 links are horizontal and 8 vertical).

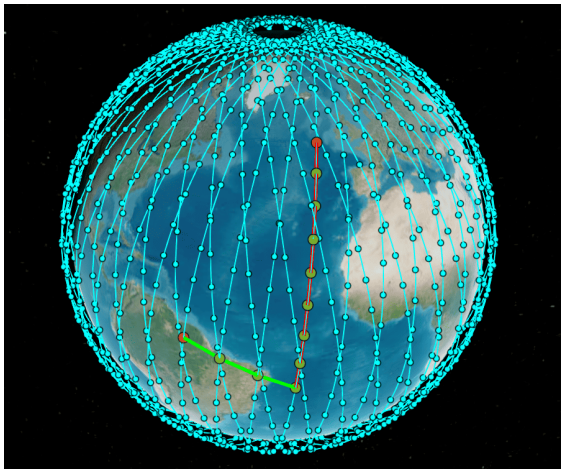


Figure 5.8: Trivial Path on VLEO 96.9°:2000/40/21 from (37,5) to (0,18)

o	i	Single ID
37	5	1856
38	5	1906
39	5	1956
0	26	27
0	25	26
0	24	25
0	23	24
0	22	23
0	21	22
0	20	21
0	19	20
0	18	19

Table 5.5: Trivial Route Path on VLEO from (37,5) to (0,18)

`FlipCoin` route is shown in Figure 5.9. Since it also uses Minimum Hop Count computation, the direction and the number of total hops (therefore, the amount of both horizontal and vertical ones) are the same as `Trivial` routing. Indeed, Minimum Hop Count shrinks the search space to a rectangular grid on the constellation, where only Manhattan distances can be crossed. However, the choice of nodes to be traversed is again dependent on the algorithms' implementation.

`Dijkstra`'s route is presented in Figure 5.10 and Table 5.7. It is the same result as `DisCoRoute` when it handles (37,5) and (18,0) as both Descending satellites (D2D case).

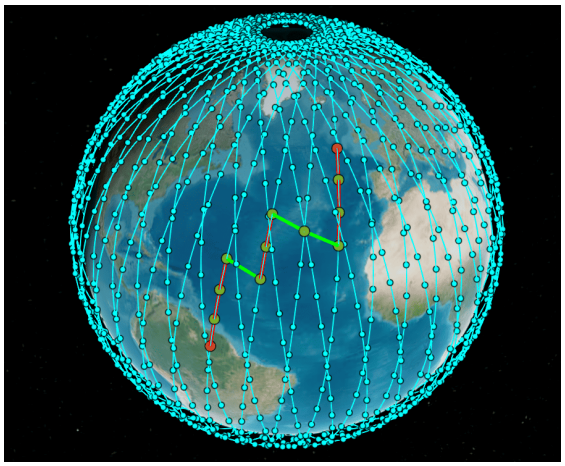


Figure 5.9: Flip Coin Path on VLEO 96.9°:2000/40/21 from (37,5) to (0,18)

o	i	Single ID
37	5	1856
37	4	1855
37	3	1854
37	2	1853
38	2	1903
38	1	1902
38	0	1901
39	0	1951
0	21	22
0	20	21
0	19	20
0	18	19

Table 5.6: Flip Coin Route Path on VLEO from (37,5) to (0,18)

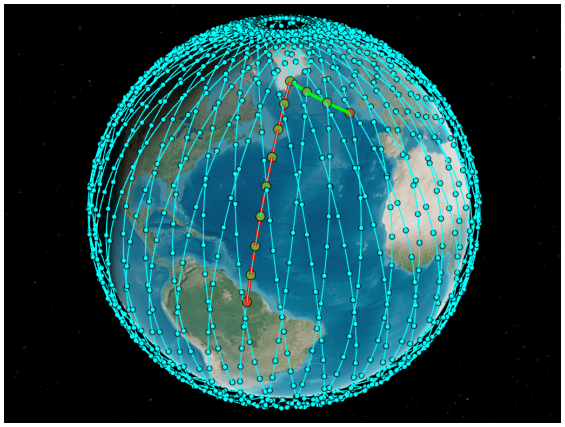


Figure 5.10: Dijkstra Path on VLEO 96.9°:2000/40/21 from (37,5) to (0,18)

o	i	Single ID
37	5	1856
37	4	1855
37	3	1854
37	2	1853
37	1	1852
37	0	1851
37	49	1900
37	48	1899
37	47	1898
38	47	1948
39	47	1998
0	18	19

Table 5.7: Dijkstra Route Path on VLEO from (37,5) to (0,18)

5.3 Performance Evaluation

In the following section the performance in terms of propagation delay and computational time is shown and commented. The results are depicted both for LEO and VLEO as usual, taking the conventional Dijkstra's algorithm as a baseline.

The propagation delay is expressed in percentage to be consistent with [5].
In Figure 5.11 the performance of `Trivial`, `FlipCoin` and `DisCoRoute` is shown for LEO 53°:1584/72/39 at 550 km over 20000 random pairs. All the algorithms are at most 3% worse than `Dijkstra`'s, however, unlike `Dijkstra`, they guarantee the Minimum Hop Count. The outcomes reproduce successfully what was proposed in [5].
Then, in Figure 5.12 the investigation of `Trivial`'s, `FlipCoin`'s and `DisCoRoute`'s propagation delay is extended to VLEO 96.9°:2000/40/21 at 360 km over 20000 random pairs. In this case, there is a clear degradation of performance since all the algorithms perform even 60% worse than `Dijkstra`, with the exception of `DisCoRoute` performing at most 9% worse. The results show that there is room of improvement in this scenario.

When observing the computational time of LEO (Figure 5.13) and VLEO (Figure 5.14), it is noticeable that in both cases `Dijkstra`'s computational time is huge with respect to `Trivial`, `FlipCoin` and `DisCoRoute`: it depends on the implementation of the algorithms and the different search space of `Dijkstra` with respect to the one computed with Minimum Hop Count.

Figures 5.15 and 5.16 are histograms that illustrate the comparison in terms of hop count between conventional `Dijkstra` and `DisCoRoute` strategies. They have been analyzed over 20000 random pairs of source and destination satellites belonging to LEO and VLEO configurations.
`DisCoRoute` prioritizes finding paths with the fewest possible hops by using the Minimum Hop Count criterion. In contrast, `Dijkstra`'s algorithm focuses on selecting paths with the smallest propagation delay, which can result in choosing routes that, while shorter, may involve a greater number of hops than those selected by DisCoRoute.
LEO Walker Delta configuration 53°:1584/72/39 has the majority (19816/20000) of paths selected by both `Dijkstra` and `DisCoRoute` with the same number of hops. The remaining 184 routes have 1 hop more when computed with `Dijkstra` instead of `DisCoRoute` (Fig. 5.15).
VLEO 96.9°:2000/40/21 has 19135 paths in which `Dijkstra` and `DisCoRoute` share the same number of hops, however it has also 361 where `Dijkstra` has 1 more hop than `DisCoRoute`, 228 with 2 more hops, up to 12 with 11 more hops (Fig. 5.16).

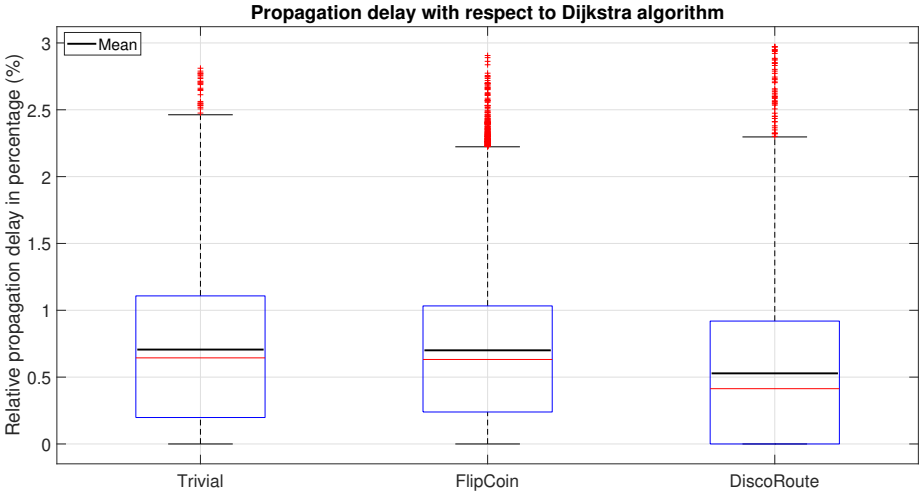


Figure 5.11: Boxplot of propagation delay in LEO 53°:1584/72/39 over 20000 random pairs

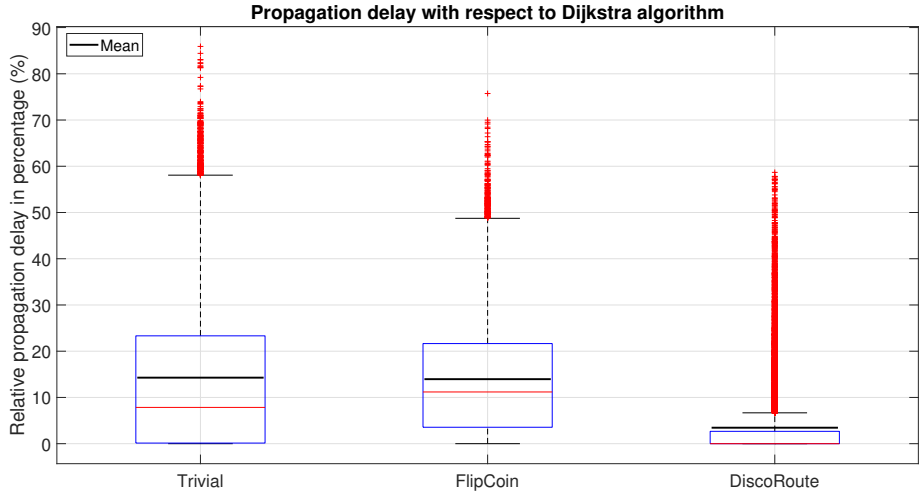


Figure 5.12: Boxplot of propagation delay in VLEO 96.9°:2000/40/21 over 20000 random pairs

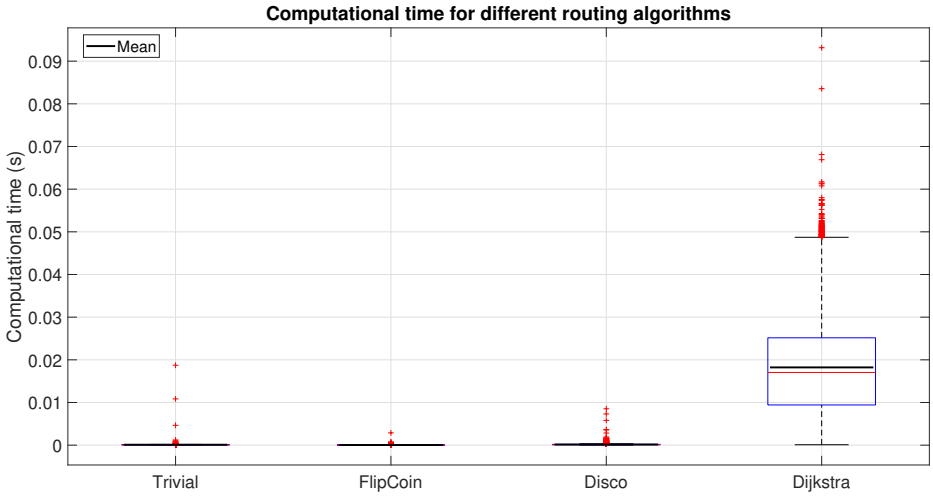


Figure 5.13: Boxplot of computational time in LEO 53°:1584/72/39 over 20000 random pairs

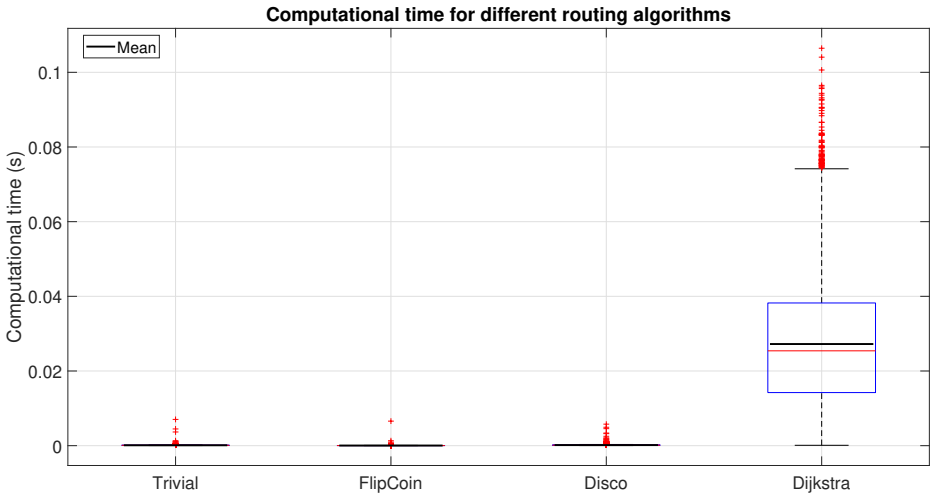


Figure 5.14: Boxplot of computational time in VLEO 96.9°:2000/40/21 over 20000 random pairs

The choice of one algorithm with respect to another should be done based on a trade-off: if the goal is the shortest distance, of course `Dijkstra` is the optimal solution. However, this strategy does not take into account the possible processing time, queueing delay and transmission time that are present at each node. If they are not negligible, then `DisCoRoute` becomes a better choice, despite being sub-optimal.

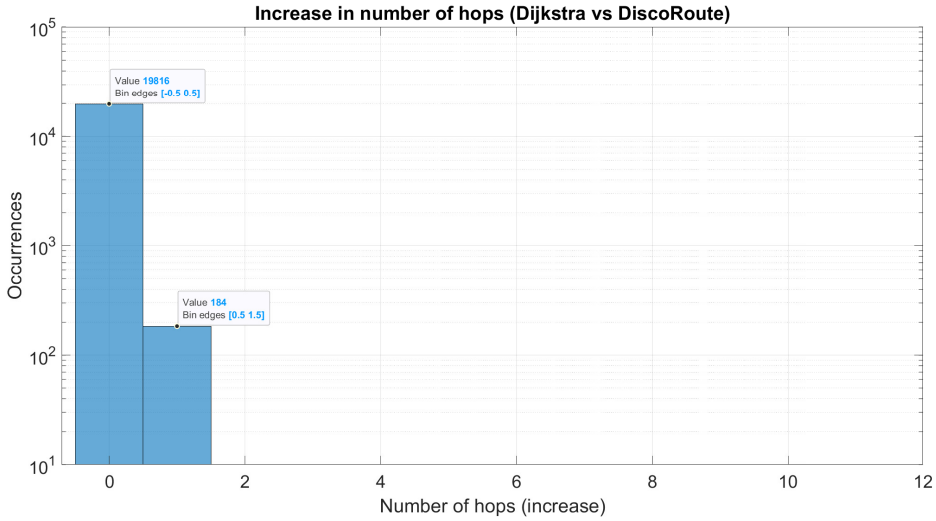


Figure 5.15: Increase in hops of Dijkstra with respect to DisCoRoute on LEO 53°:1584/72/39 over 20000 random pairs

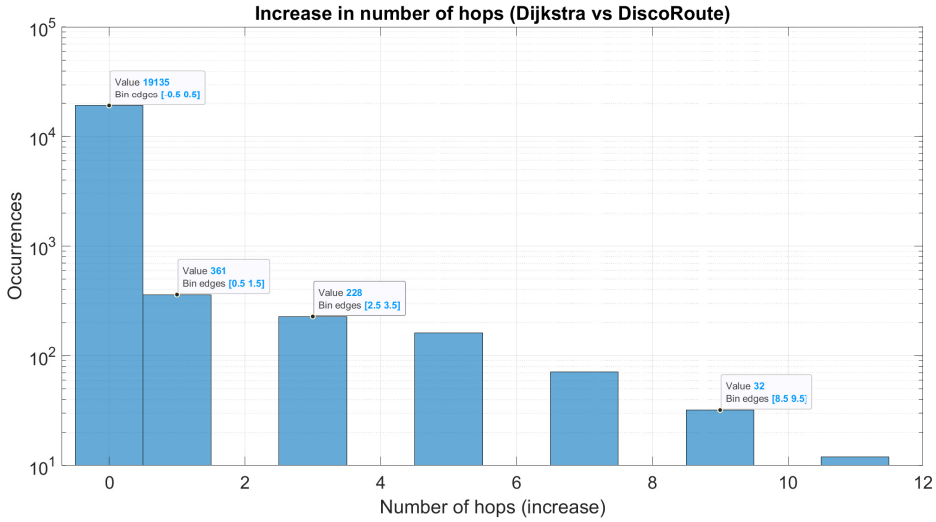


Figure 5.16: Increase in hops of Dijkstra with respect to DisCoRoute on VLEO 96.9°:2000/40/21 over 20000 random pairs

5.4 Constellation 2D grid

Figures 5.17 and 5.18 visualize the same constellation patterns analyzed so far, focusing on a 2D representation where the rows correspond to the orbit plane indices (o) and the columns are the labels (i) of the satellites within an orbit (o). The result is a satellite 2D grid, with each node represented by a unique pair of indices (o,i). Then, the distances between every satellite and its four neighbours are computed and normalized with respect to the global maximum distance (so that their value goes from 0 to 1). Noteworthy is the fact that their normalized magnitude is associated with a color map to show their impact in the constellation.

On the one hand, LEO configuration present distances that have similar colours (corresponding to the nuances of orange in Figure 5.17). This implies that they are close to each other in their magnitude and also analogous to the maximum distance (which is dark red).
On the other hand, VLEO's 2D grid shows a significant change regarding some areas, where the distances are noticeably shorter (indeed, they are represented with colours like yellow, light blue and blue). These zones correspond to the Arctic and Antarctic regions.

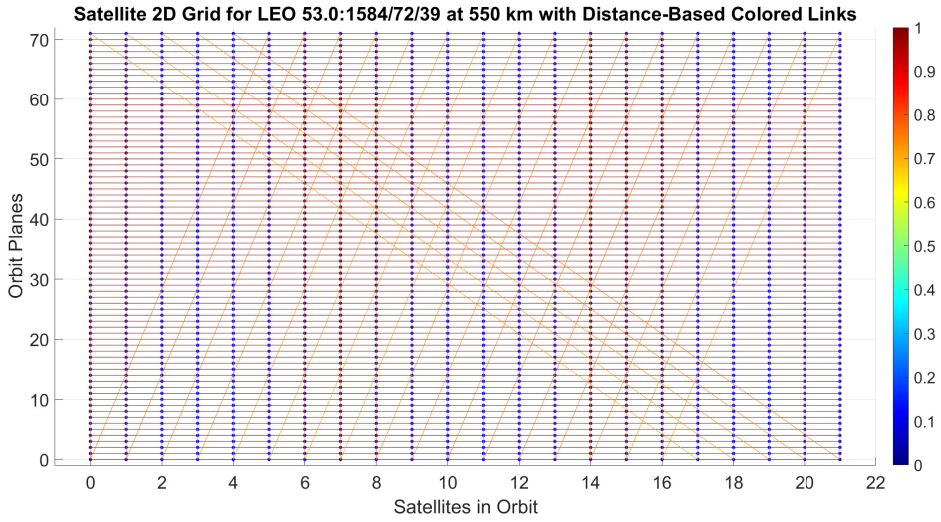


Figure 5.17: Constellation 2D grid for LEO 53°:1584/72/39

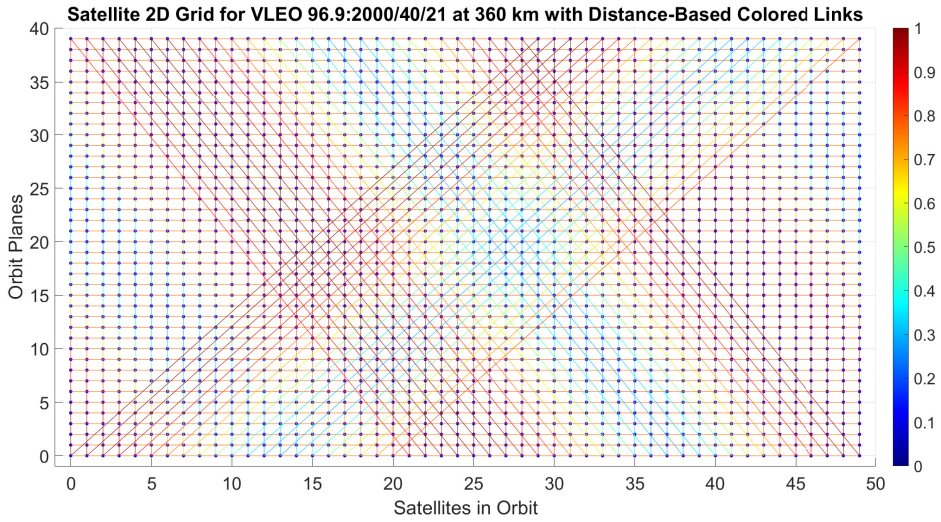


Figure 5.18: Constellation 2D grid for VLEO 96.9°:2000/40/21

5.5 Routing analysis changing altitude, inclination and topology

The analysis of the two-dimensional representations of the constellation designs for LEO and VLEO reveals notable changes in the topology near the polar regions. Shorter distances between neighbouring satellites may produce also variations in the performance of the algorithms. Especially, `DisCoRoute` constructs segments of its final routing path by using Minimum Hop Count and the latitudes of source and destination satellites. To further analyze the performance of these algorithms, tests are conducted by varying one parameter at a time, inclination, altitude, or topology (defined by the number of planes and the number of satellites per plane), while keeping the other two parameters fixed. This approach allows for a detailed evaluation of how each parameter influences algorithmic performance. Several simulations have been launched over 2000 random pairs to observe routing performance in terms of average end-to-end delay (hereafter referred to as latency) among pairs with specified benchmark values for altitude, inclination, topology and transmission time.
The chosen quantities are:

- altitude: 360 km, 550 km, 800 km, 1000 km,
- inclination: 53°, 96.9°, 105°, 130°,
- topology: 1584/72/21 and 2000/40/39,
- transmission time: from 0 to 100ms.

5.5.1 Variation of altitude

The benchmarks (360 km, 550 km, 800 km, 1000 km) have been selected both based on the technical reports of [32] and [33] and empirically.

Distribution of distances and performance of algorithms for configuration 53°:1584/72/39

The topology of the constellation is 1584/72/39, the inclination is fixed at 53° and the transmission time at zero. The altitude varies from 360 km to 1000 km.
Figure 5.19 represents the distribution of vertical and horizontal distances for the just mentioned scenario. As expected, the higher the altitude, the larger the magnitude of the distances. It's quite interesting to note that the distribution of horizontal hops slightly increases with altitude as well.
Figures 5.20 and 5.21 both depict the relation between end-to-end delay and altitude. The difference is that the second figure takes `DijkstraTrans` as baseline (with transmission time equal to zero). In line with intuitions, the latency increases with altitude and the mutual performance of algorithms is almost unchanged, with `Dijkstra`, `DijkstraTrans` being optimal and `DisCoRoute` being better than `Trivial` and `FlipCoin`.

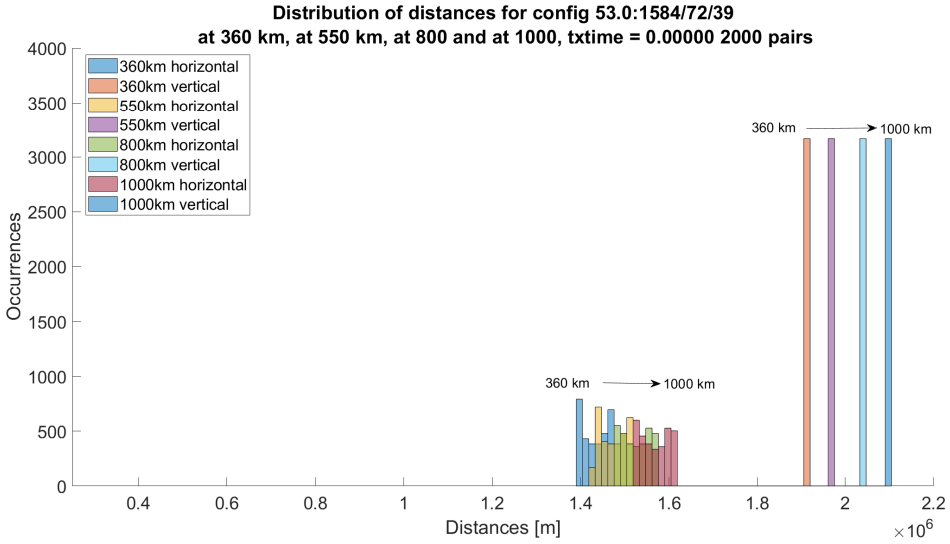


Figure 5.19: Distribution of distances in 53°:1584/72/39 changing altitude

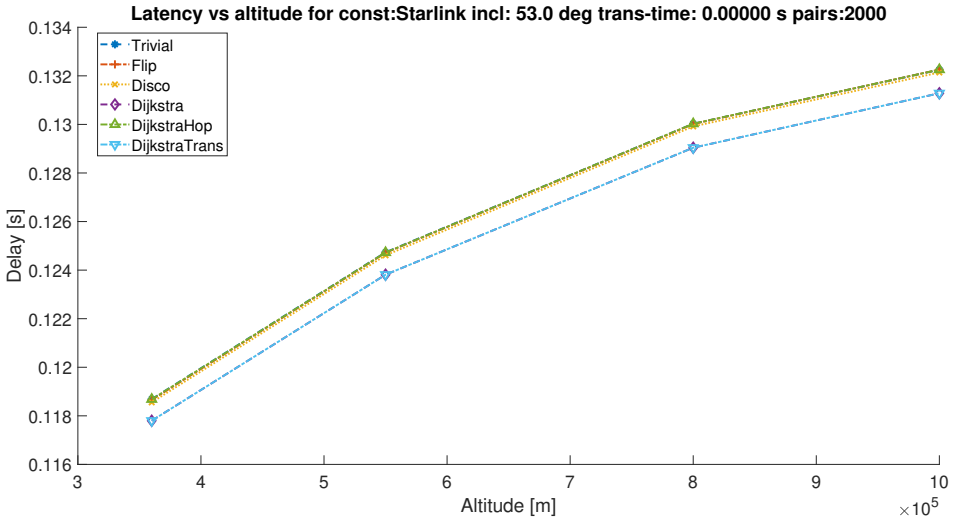


Figure 5.20: Latency vs Altitude for 53°:1584/72/39 over 2000 random pairs

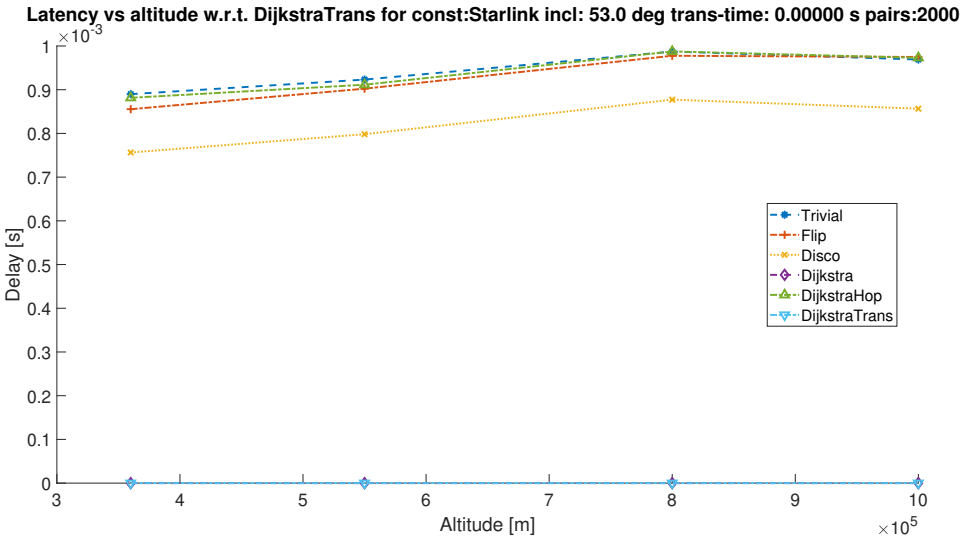


Figure 5.21: Latency vs Altitude w.r.t. DijkstraTrans for 53°:1584/72/39 over 2000 random pairs

Distribution of distances and performance of algorithms for configuration 96.9°:2000/40/21

Analogously to the previous case, the configuration of the topology is 2000/40/21, the inclination is 96.9° and the transmission time is set to zero. The altitude changes from 360 km to 1000 km.

In Figure 5.22 the distribution of vertical and horizontal distances for the chosen layout is shown. The same considerations done for the previous case are also valid here. Notably, the distribution of horizontal hops exhibits a marked increase as altitude rises. Selecting the appropriate horizontal hop makes a substantial difference in terms of distance, and consequently, in delay.
In Figures 5.23 and 5.24 similar observations to the 1584/72/39 case hold here as well.

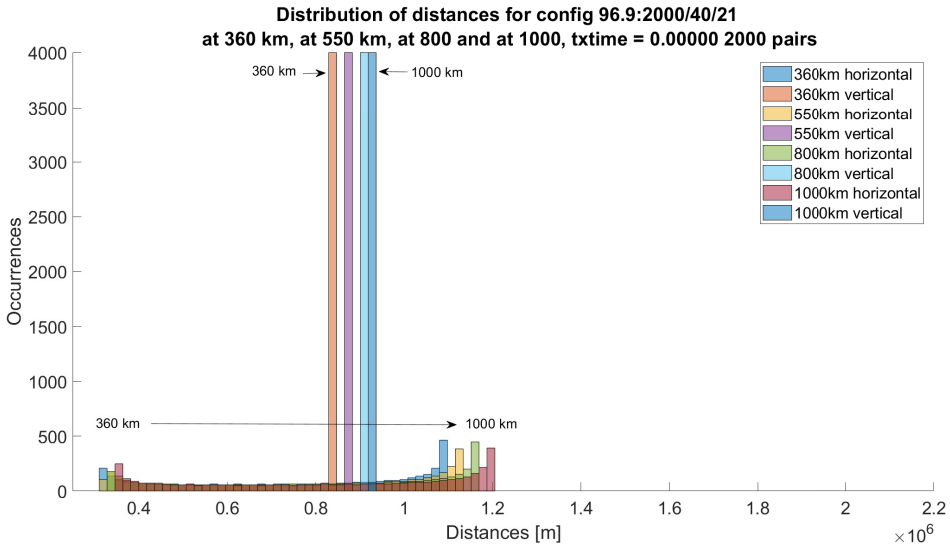


Figure 5.22: Distribution of distances in 96.9°:2000/40/21 changing altitude

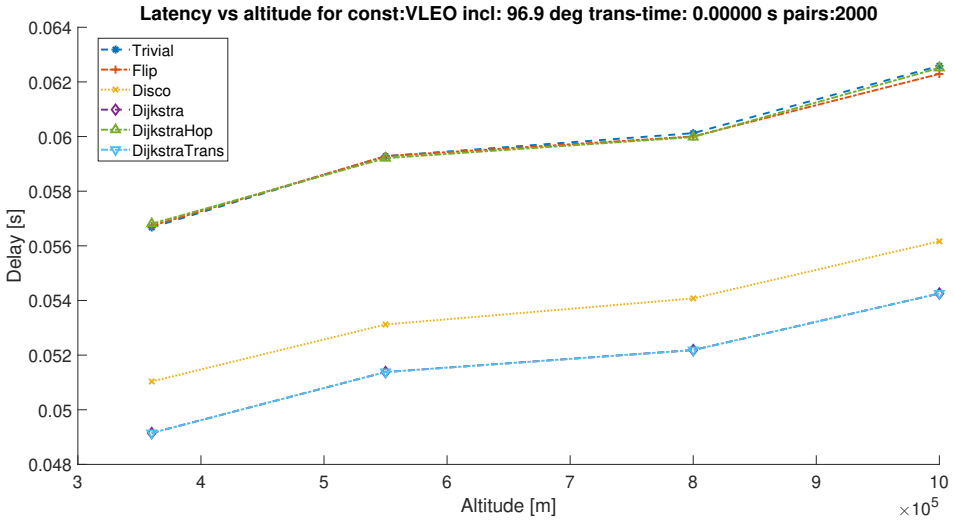


Figure 5.23: Latency vs Altitude for 96.9°:2000/40/21 over 2000 random pairs

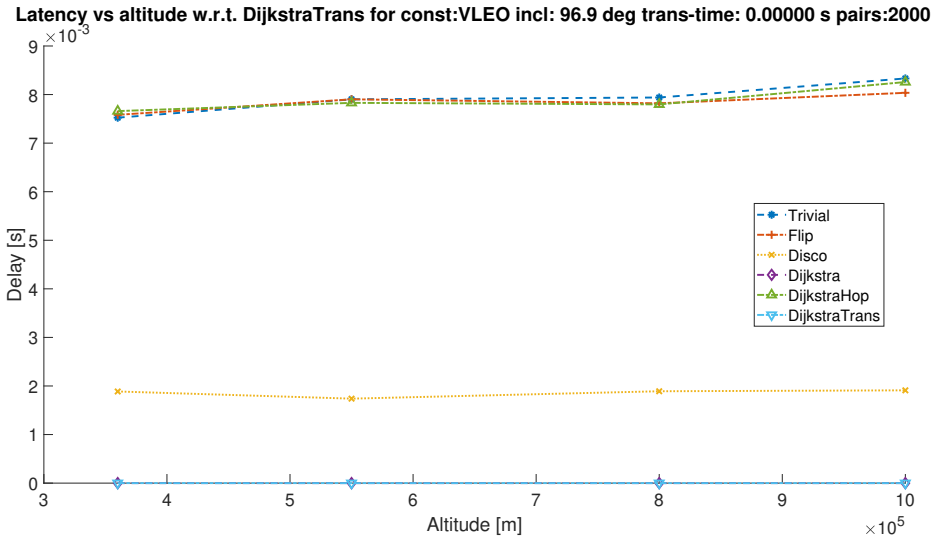


Figure 5.24: Latency vs Altitude w.r.t. DijkstraTrans for 96.9°:2000/40/21 over 2000 random pairs

5.5.2 Variation of inclination

The benchmarks (53°, 96.9°, 105°, 130°) have been chosen based on the technical reports of [32] and [33].

Distribution of distances and performance of algorithms for configuration 1584/72/39

The constellation pattern is 1584/72/39, the altitude is fixed at 550 km and the transmission time at zero. The inclination varies from 53° to 130°.

The distribution of vertical and horizontal distances is in Figure 5.25. Interestingly, the higher the inclination, the smaller the magnitude of the distances. Noteworthy the fact that the distribution of horizontal hops slightly increases with inclination as well.

Figures 5.26 and 5.27 show the behaviour of the end-to-end delay with respect to inclination. As usual, the second figure takes `DijkstraTrans` as baseline (with transmission time equal to zero). The latency decreases with inclination. Further considerations on the performance of the algorithms are presented in the 2000/40/21 case.

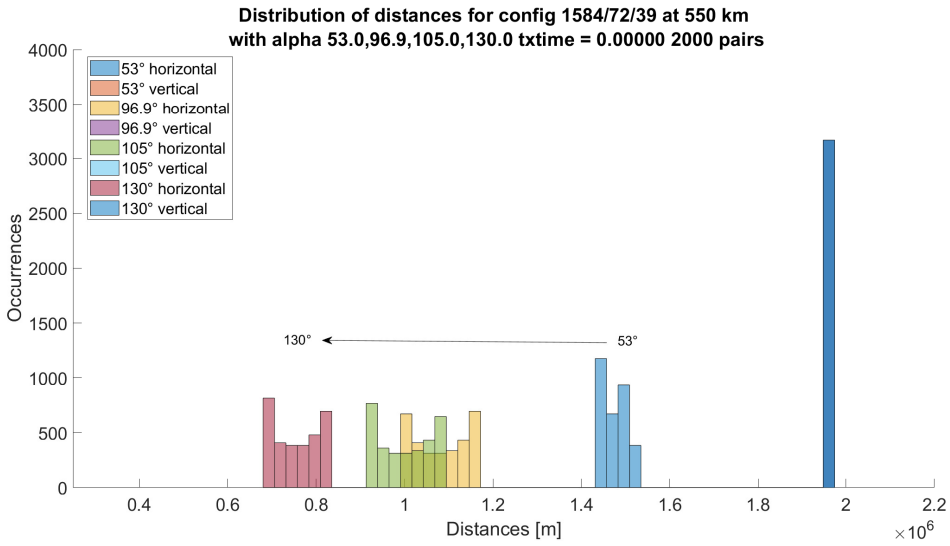


Figure 5.25: Distribution of distances in 1584/72/39 changing inclination

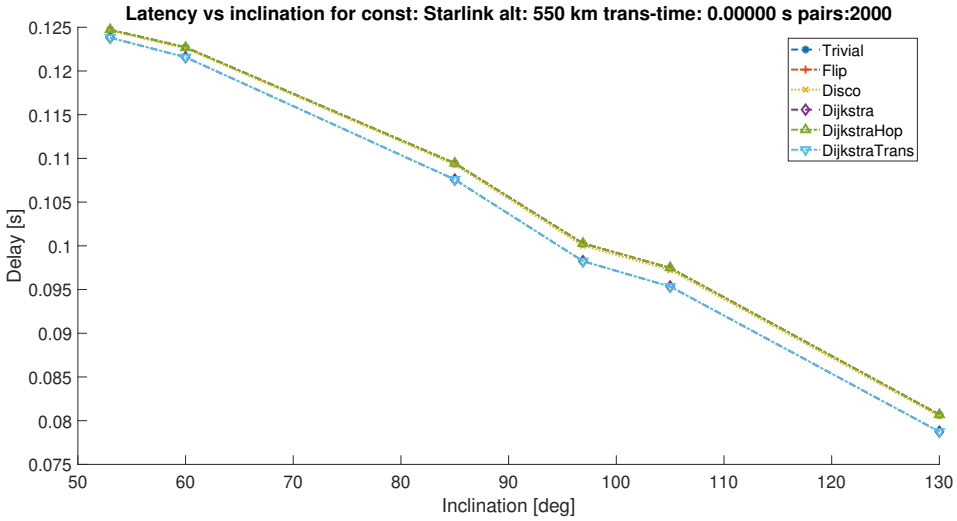


Figure 5.26: Latency vs Inclination for 1584/72/39 over 2000 random pairs

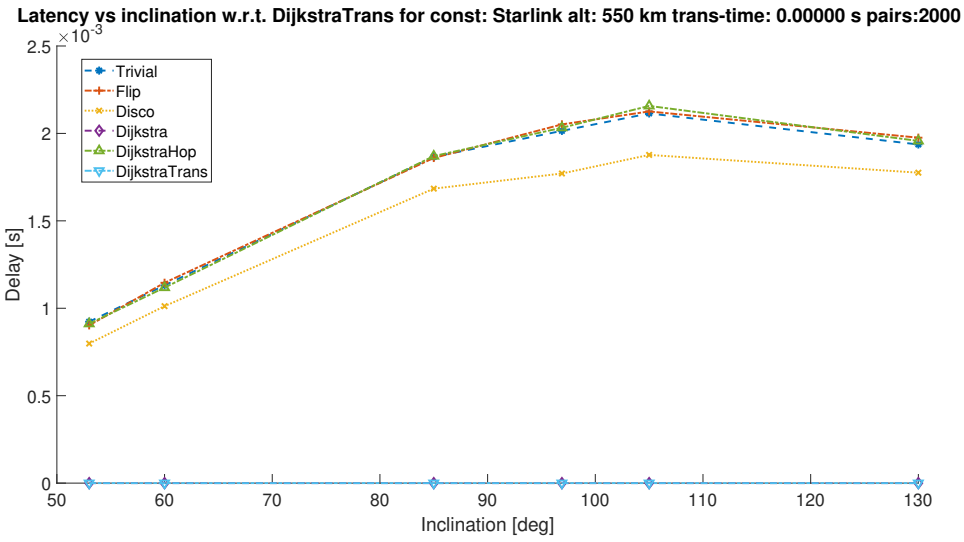


Figure 5.27: Latency vs Inclination w.r.t. Transmission Time for 1584/72/39 over 2000 random pairs

Distribution of distances and performance of algorithms for configuration 2000/40/21

In the following section, the configuration of the constellation under test is 2000/40/21, the altitude is set at 360 km and the transmission time at zero. The inclination varies from 53° to 130°.

The distribution of vertical and horizontal distances is in Figure 5.28. As for the previous case, the distances get smaller in magnitude the higher the inclination. In this scenario, the distribution of horizontal hops significantly grows with inclination. The latency with respect to the inclination is described as usual by Figures 5.29 and 5.30. All the algorithms perform better by increasing the inclination. In particular, `Dijkstra`, `DijkstraTrans` and `DisCoRoute` perform better the higher the inclination.

The latency decreases, however it suggests either a stabilization point or an increase the higher the inclination. Therefore, a further study has been conducted including the full period of possible inclinations from 0° to 360°.
Figures 5.31 and 5.32 show the results. The second figure collects the same outcomes, however it represents them taking `DijkstraTrans` as a baseline. What emerges is that there is a symmetry in performance around 180°, which probably depends on the Walker Delta initial set-up. Indeed, the findings in Figure 5.29 correspond exactly to the interval from 53° to 130° of Figure 5.31.

Figure 5.33 is illustrated in order to correlate the physical properties of the constellation with the resulting performance in routing. It is the plot of the standard deviation of horizontal hops (Distances in [m]) in the previously shown histograms of 2000/40/21 when changing inclination and the average delay (Delay in [s]) of `DisCoRoute` taking as a baseline `DijkstraTrans`. Both are normalized with respect to their maximum value.
Interestingly, Pearson correlation between delays and distances in the interval from 30° to 130° is very high (0.8874).

As a final insight, `DisCoRoute` and Minimum Hop Count work well when hops have similar distances so that minimizing the number of hops is almost equivalent to computing the total shortest distance (Dijkstra).

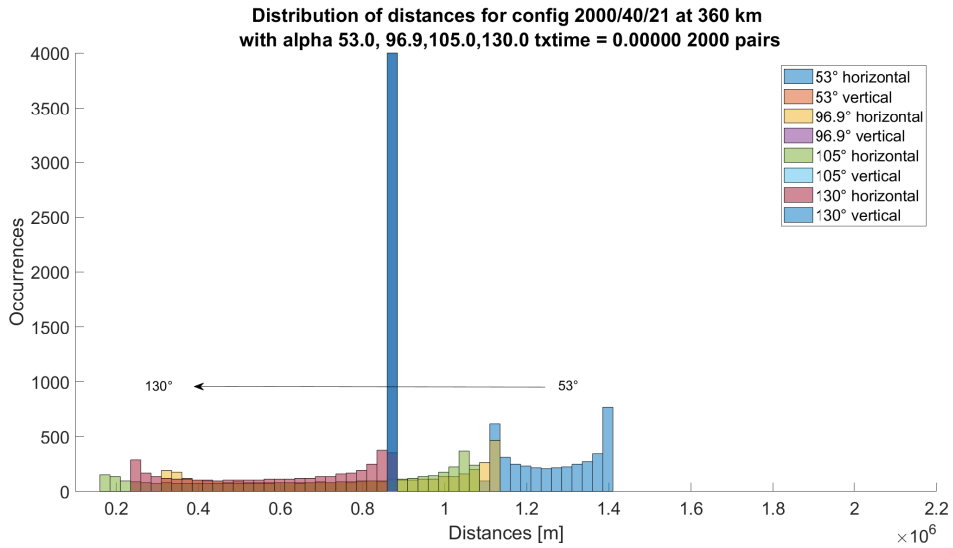


Figure 5.28: Distribution of distances in 2000/40/21 varying inclination

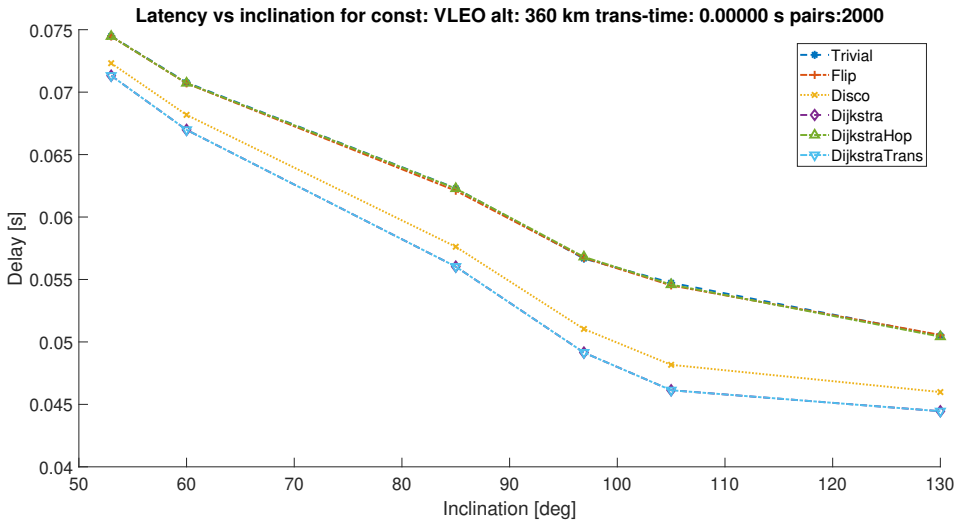


Figure 5.29: Latency vs Inclination for 2000/40/21 over 2000 random pairs

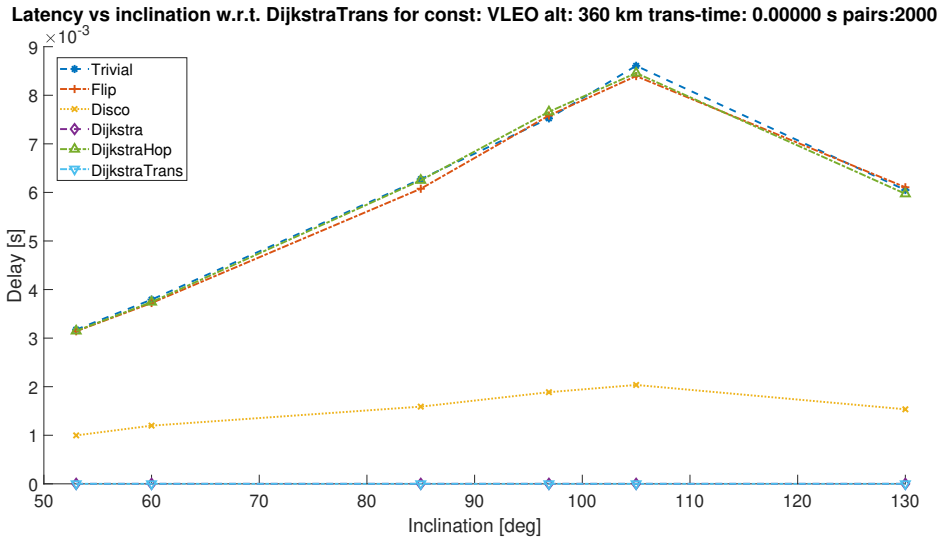


Figure 5.30: Latency vs Inclination w.r.t. DijkstraTrans for 2000/40/21 over 2000 random pairs

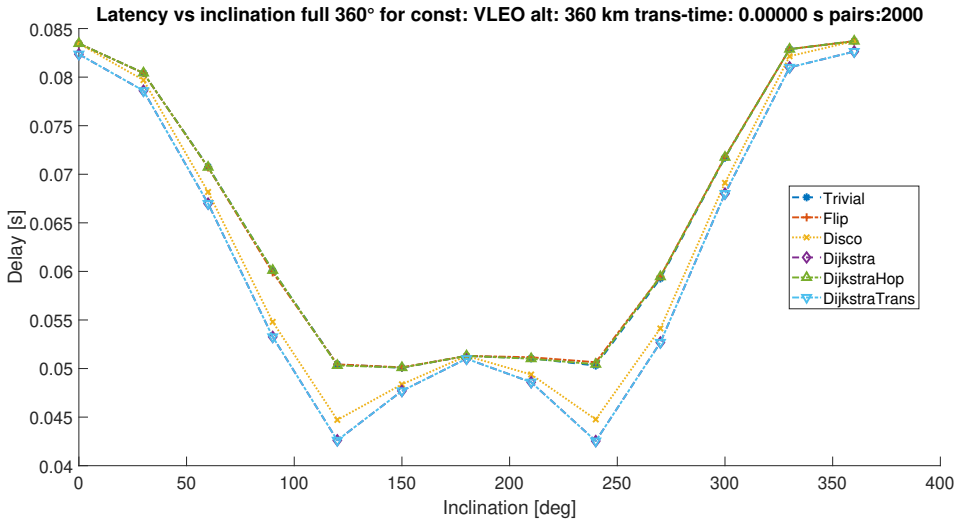


Figure 5.31: Latency vs Inclination full 360° for 2000/40/21 over 2000 random pairs

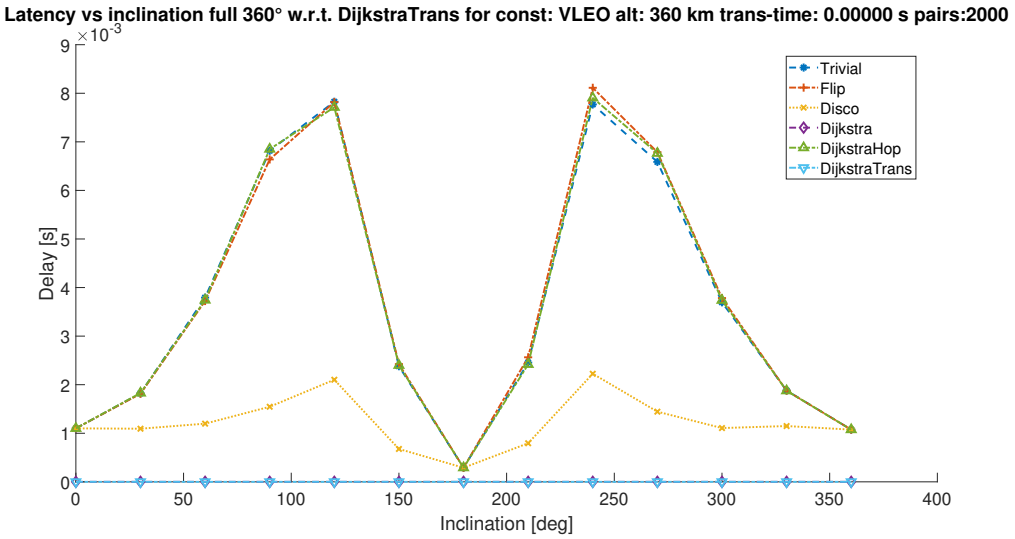


Figure 5.32: Latency vs Inclination full 360° w.r.t. DijkstraTrans for 2000/40/21 over 2000 random pairs

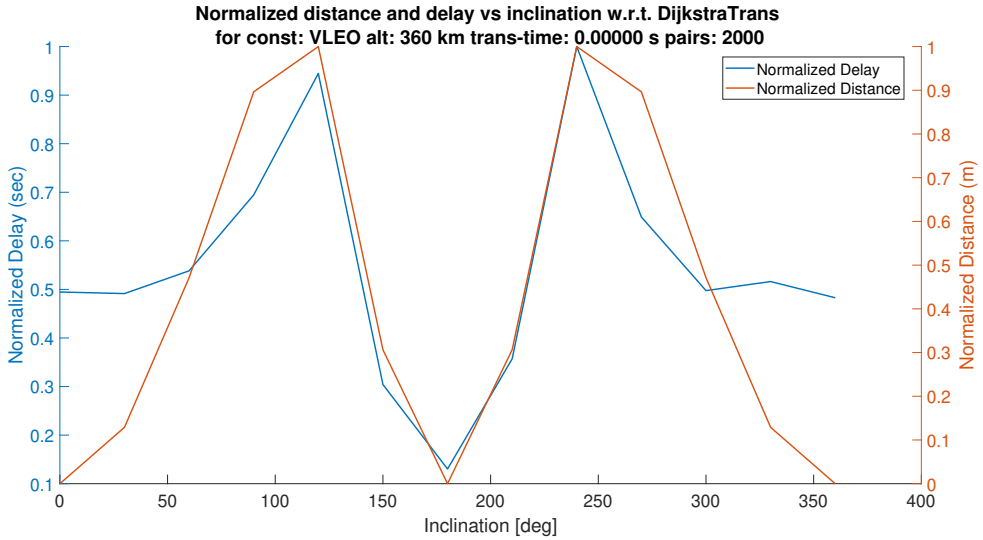


Figure 5.33: Normalized distance and delay vs Inclination full 360° w.r.t. DijkstraTrans for 2000/40/21 over 2000 random pairs

5.5.3 Variation of topology

Selected topologies are 1584/72/39 and 2000/40/21. Figure 5.34 and Figure 5.35 present two cases the first with same inclination 53° and altitude 550 km and the second with inclination 96.9° and altitude 550 km. In the case of 1584/72/39, utilizing vertical hops (the red bar) is not convenient due to their length, which can lead to inefficiencies. Conversely, the situation is reversed in 2000/40/21, where vertical hops are more advantageous since they are shorter than several horizontal hops. Notably, the distribution of horizontal hops in 2000/40/21 is larger than that in 1584/72/39. This emphasizes the critical importance of selecting the right horizontal hops in 2000/40/21, as this decision has a significant impact on overall performance.

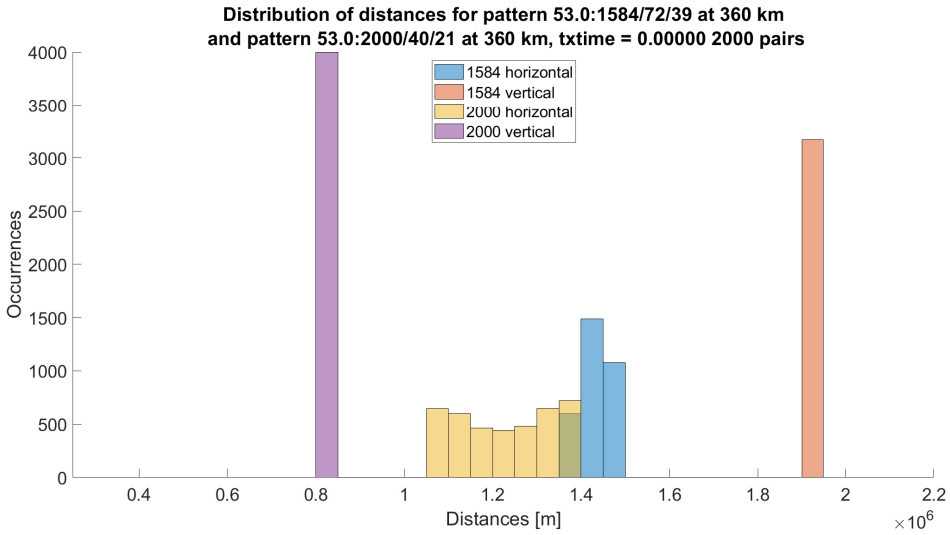


Figure 5.34: Same inclination at 53°, same altitude at 360 km, different topologies 2000/40/21 and 1584/72/39

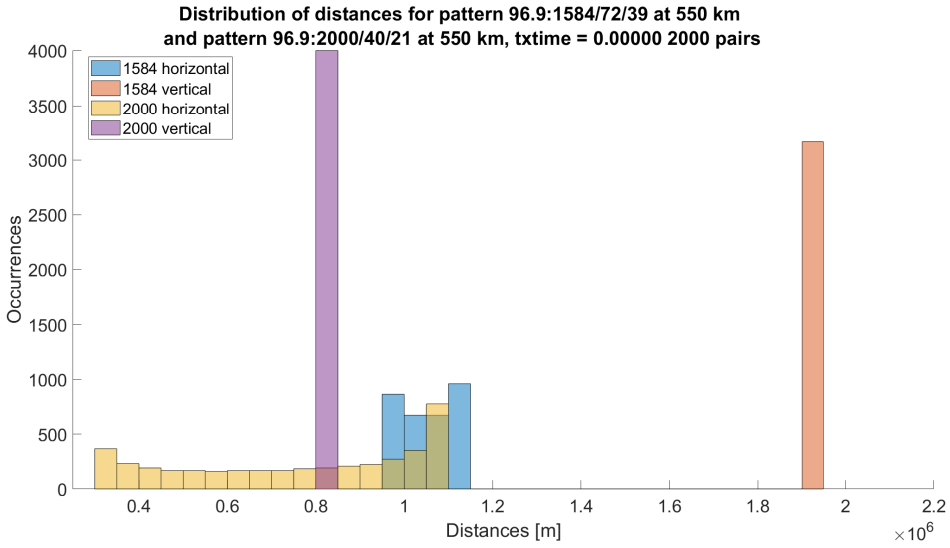


Figure 5.35: Same inclination at 96.9°, same altitude at 550 km, different topologies 2000/40/21 and 1584/72/39

5.5.4 Performance changing the transmission time

The transmission time goes from 0 to 100ms. These values have been decided to be consistent with the maximum latency between two neighbouring satellites (for instance in 2000/40/21 is 3.7 ms). Figures 5.36 and 5.37 represent the traditional 53°:1584/72/39 setup taking `DijkstraTrans` as a baseline.
Figures 5.38 and 5.39 represent the 96.9°:2000/40/21 pattern, taking the usual `DijkstraTrans` as a baseline.

The following observations are presented and hold for all the just-mentioned scenarios:

- `DijkstraTrans` performs better than `Dijkstra` when the transmission time is high (because it selects a better route with fewer hops when transmission time is high)
- At a certain time even `DisCoRoute` performs better than `Dijkstra` because it has the minimum number of hops calculated with Minimum Hop Count (and for each hop the transmission time is added). This phenomenon is even more evident in Figure 5.40.

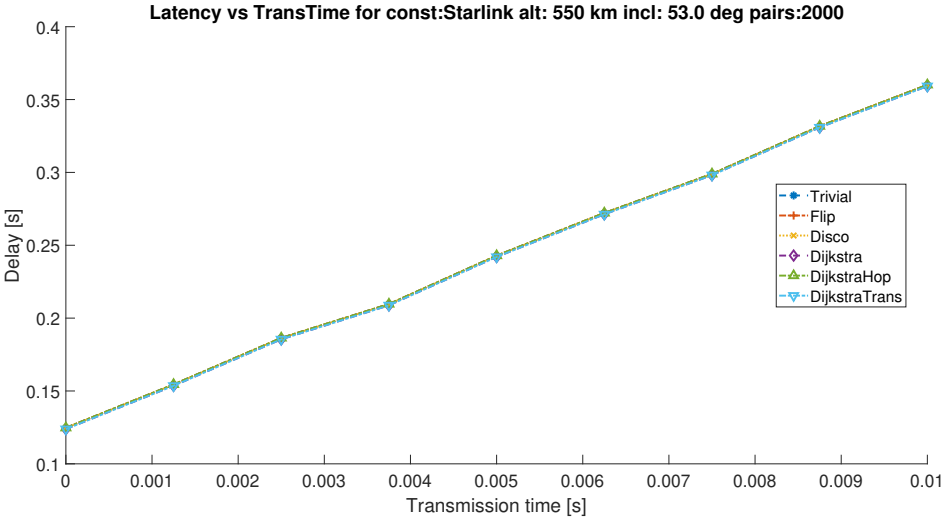


Figure 5.36: Latency vs Transmission Time 53°:1584/72/39 at 550 km over 2000 random pairs

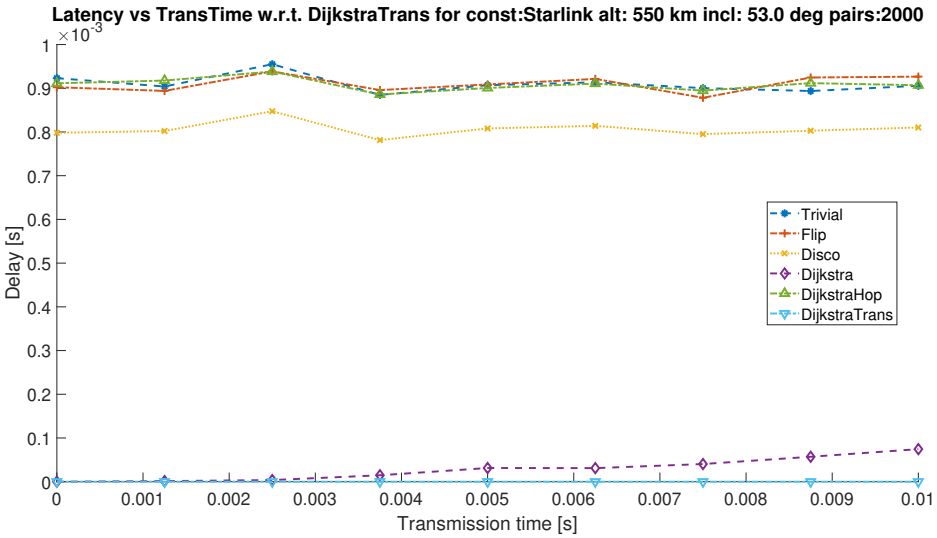


Figure 5.37: Latency vs Transmission Time w.r.t DijkstraTrans 53°:1584/72/39 at 550 km over 2000 random pairs

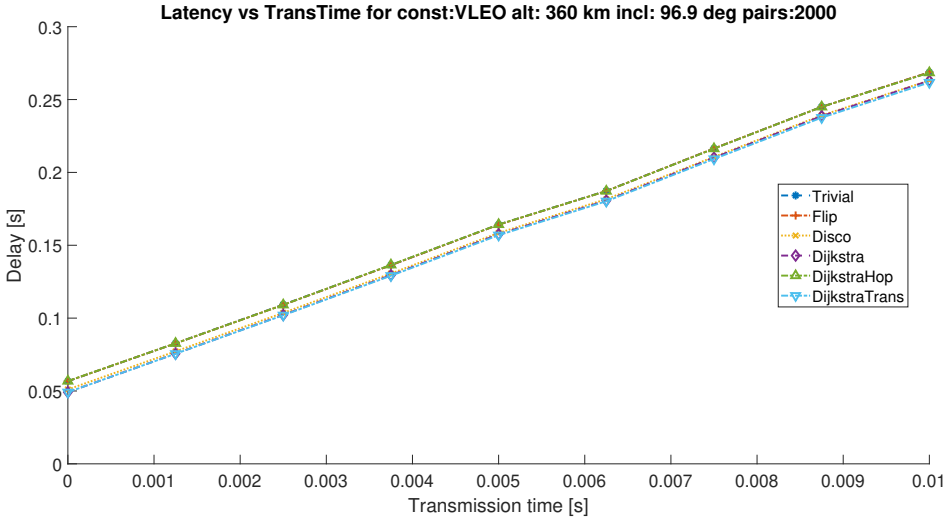


Figure 5.38: Latency vs Transmission Time 96.9°:2000/40/21 at 360 km over 2000 random pairs

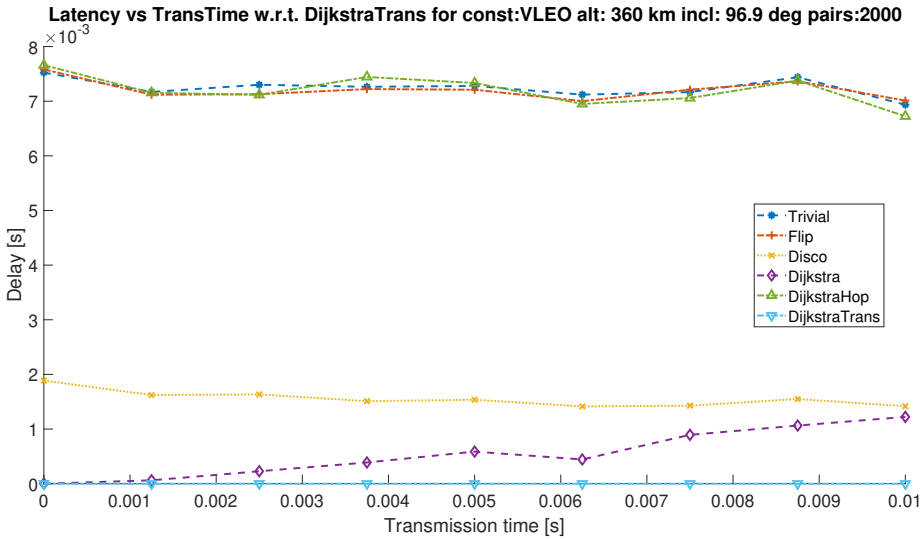


Figure 5.39: Latency vs Transmission Time w.r.t. DijkstraTrans 96.9°:2000/40/21 at 360 km over 2000 random pairs

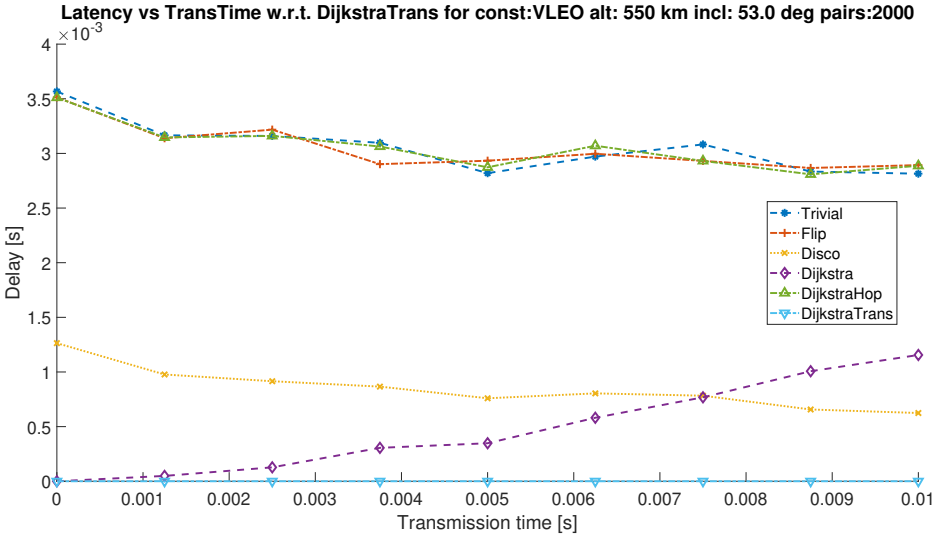


Figure 5.40: Latency vs Transmission Time 53°:2000/40/21 at 550 km over 2000 random pairs

5.6 Enhancement: new heuristic

This section provides several illustrations of 3D plots and 2D satellite grid representations for a VLEO constellation characterized by an inclination of 96.9° and an orbital configuration of 2000/40/21. The visualizations include the orbital paths and inter-satellite connections for a specific satellite pair within this constellation. Indeed, as an example, the source satellite is identified by (34,23), while the destination satellite is (9,29). These plots effectively highlight the relative positions and movement patterns in both three-dimensional space and a flattened grid layout, showing the routing paths between the selected satellites, using the algorithms `DisCoRoute`, `Dijkstra` and the new heuristic `TAAT`.

It is evident from Figures 5.42 and 5.41 that if `DisCoRoute` is used, the potential benefits of shorter link distances in the polar regions are not fully utilized as expected. In the 2D grid representation, the selected route avoids traversing the light blue area of the grid, indicating that these regions, which could offer more efficient connections due to shorter link magnitudes, are not being adequately exploited.

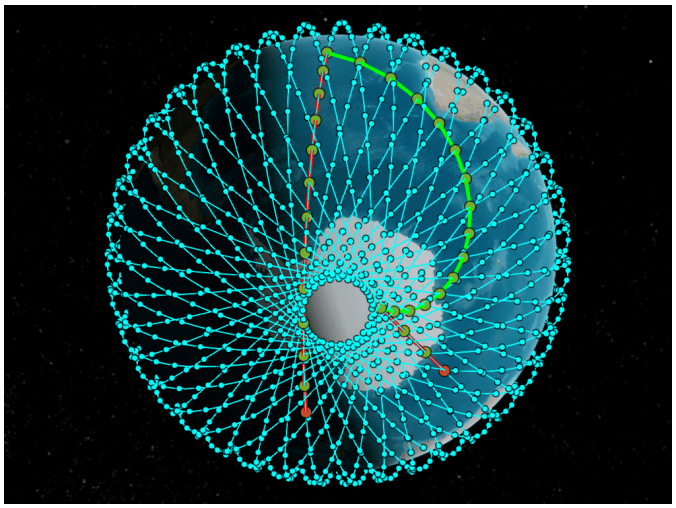


Figure 5.41: DisCoRoute Route Path on VLEO 96.9°:2000/40/21 from (34,23) to (9,29)

When employing `Dijkstra`, the satellite pair is connected thanks to a *zig-zag* route crossing the polar zones, which is clearly visible in the 3-dimensional representation from Figures 5.43 and 5.44.
The zoom has been captured making the `satelliteScenario` progress in time solely for visualization purposes, indeed, the analysis of this thesis remains static.

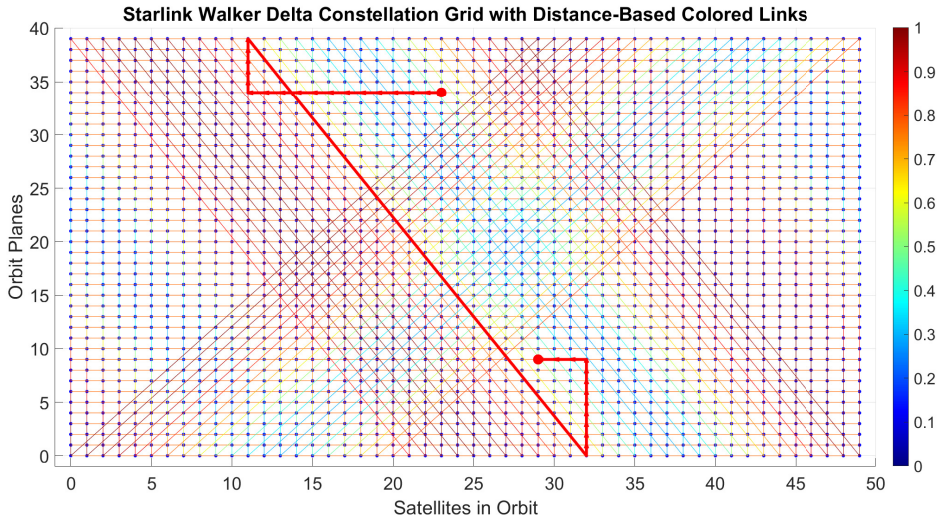


Figure 5.42: VLEO 2D grid with DisCoRoute Route Path from (34,23) to (9,29)

The *zig-zag* pattern is even more apparent in the flattened 2-dimensional representation (Figure 5.45), where the path passes through the light blue area, corresponding to the South Pole region.

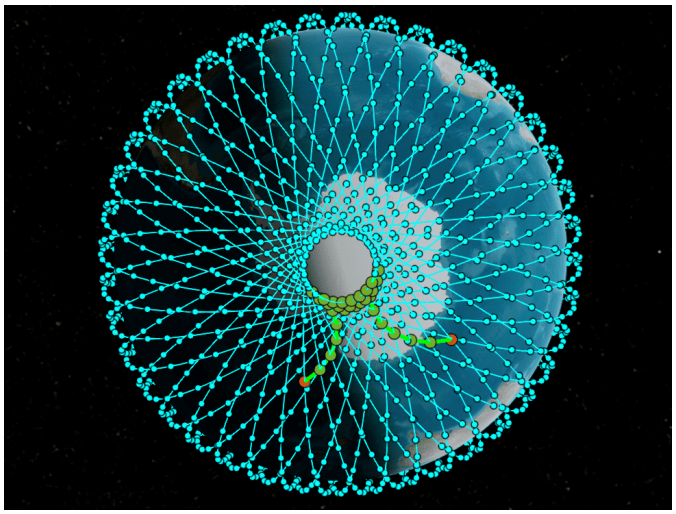


Figure 5.43: Dijkstra Route Path on VLEO 96.9°:2000/40/21 from (34,23) to (9,29)

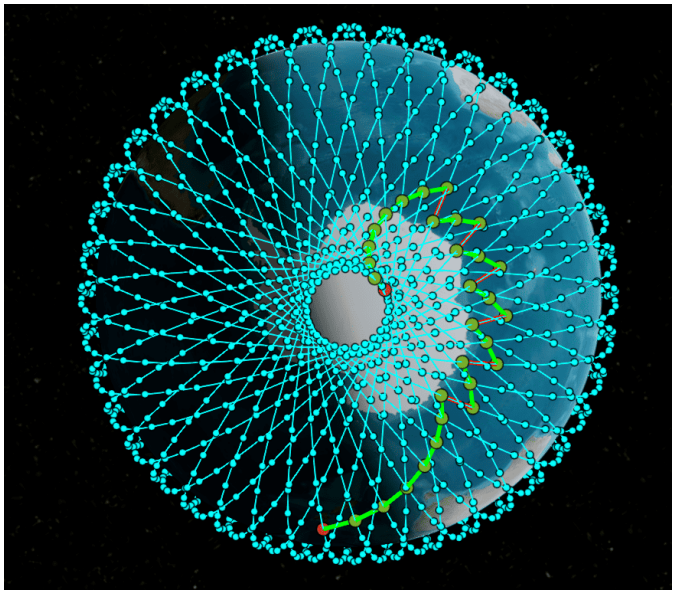


Figure 5.44: Zoom on Dijkstra Route Path from (34,23) to (9,29)

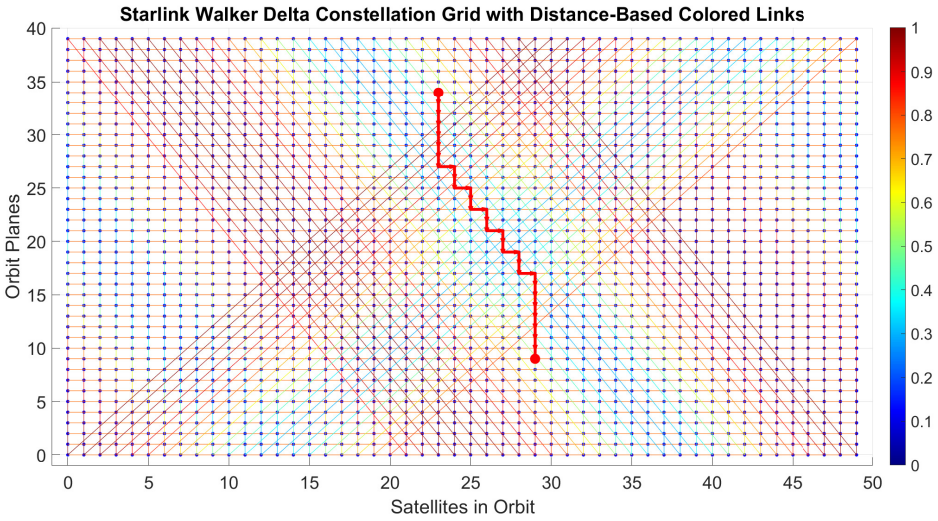


Figure 5.45: VLEO 2D grid with Dijkstra Route Path from (34,23) to (9,29)

In Figures 5.46 and 5.47 the new heuristic `TAAT` (*Targeted for Arctic and Antarctic Tracking* aims at using `DisCoRoute` outside the Arctic and Antarctic zones, whereas it constructs a 3-segments route within these regions, 2 using `Trivial` Routing and 1 connecting the polar source and polar destination employing Weighted Round Robin, reproducing a *zig-zag* route within the polar areas.

Testing the performance for the aforementioned algorithms on 2000 random pairs, the outcomes show that:

- when computing the performance over 2000 random pairs, the new heuristic is used 41 times (2% of cases),
- average delay of `TAAT` is 0.05123 s, with standard deviation equal to 0.01894s
- average delay of `DisCoRoute` is 0.05133 with standard deviation = 0.01894 s
- the average number of hops in `TAAT` is 21.3895
- the average number of hops in `DisCoRoute` is 21.3620
- the average number of hops in `Dijkstra` is 21.5010

Consequently, the `TAAT` algorithm demonstrates better performance compared to `DisCoRoute` in terms of average delay, achieving lower mean latency across the evaluated 2000 random pairs.
Additionally, it outperforms the `Dijkstra` algorithm with a reduced average number of hops, indicating more efficient routing. However, despite these advantages, the `TAAT` algorithm still tends to utilize a higher number of hops in the polar regions with respect to the minimum, due to the 3-segment strategy. This tendency suggests that there is still significant potential for improving the algorithm, particularly in the way it manages paths in areas where satellite connectivity is dense but link distances are shorter.
Such improvements could include dynamic adjustments based on regional link density, enhanced path selection criteria to prioritize shorter links in high-latitude zones. These enhancements would help in further reducing latency and improving the overall efficiency of satellite communication in VLEO constellations.

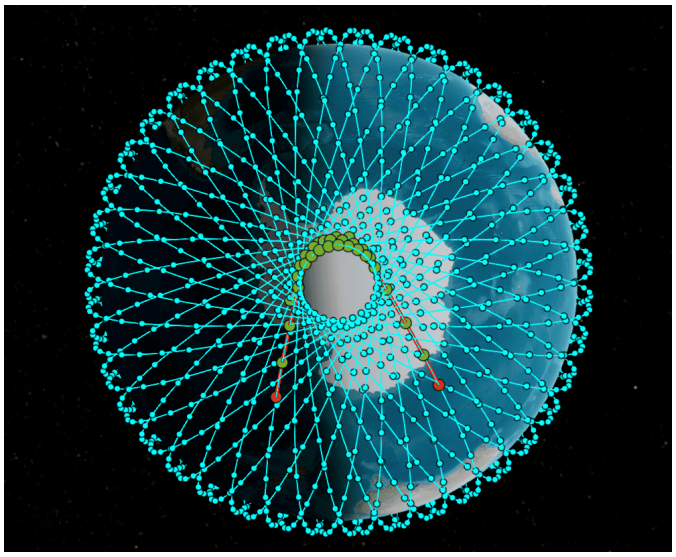


Figure 5.46: TAAT Route Path on VLEO 96.9°:2000/40/21 from (34,23) to (9,29)

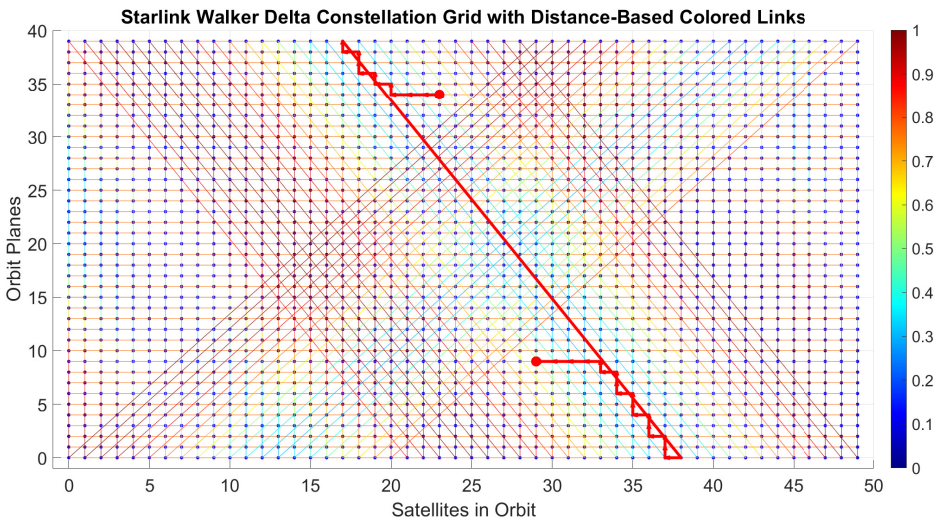


Figure 5.47: VLEO 2D grid with TAAT Route Path from (34,23) to (9,29)

Chapter 6

Conclusions and Future Works

This master's thesis is a comprehensive study of the performance of several routing strategies, initially applied on a LEO Walker Delta configuration and subsequently on a VLEO pattern.
The analysis, realized with different topologies, inclinations, and altitudes, has revealed key insights into the effectiveness of these algorithms under varying orbital conditions. One of the major findings has been the identification of a critical performance issue in DisCoRoute, the routing algorithm most suitable for Starlink's LEO 53°:1584/72/39 configuration. In particular, a performance degradation around the Arctic and Antarctic regions has been observed and consequently tackled by the introduction of a new heuristic, whose name has been defined as *TAAT* which stands for *Targeted for Arctic and Antarctic Tracking*.

Firstly, the LEO 53°:1584/72/39 at 550 km has been implemented in MATLAB. On the just-mentioned constellation, the reproducibility of the performance of `DisCoRoute` algorithm with respect to Dijkstra introduced in *Distributed On-Demand Routing for LEO Mega-Constellations: A Starlink Case Study* by G.Stock, J. Fraire and H. Hermanns [5] has been successfully tested and confirmed.

Then, the investigation has been extended to a more challenging scenario, the configuration 96.9°:2000/40/21 at 360km (VLEO). The noticeable difference in the results has suggested a drop in performance of `DisCoRoute` around the polar regions. This fact has prompted a further analysis, that has been conducted by changing the altitude (selecting as benchmark values 360 km, 550 km, 800 km, 1000 km), the inclination (with 53°,60°,85°,105°,130°) and the topologies (different number of planes and number of satellites per plane). The outcomes have shown an increase in latency (propagation and transmission delays) proportional to the altitude and inversely proportional to the inclination (due to the symmetry of the Walker Delta configuration around 180°).

Finally, the behaviour of `DisCoRoute` algorithm in the Arctic and Antarctic regions has been addressed by proposing the new heuristic (TAAT). The latter selects `DisCoRoute` whenever both source and destination are far from the polar areas, otherwise it computes a route composed of 3 segments, 2 of which use Trivial Routing and 1 Weighted Round Robin. This hybrid approach has led to improved performance, particularly in reducing average latency.

In conclusion, this thesis has contributed a novel routing heuristic, TAAT, that successfully addresses the critical performance issues of existing algorithms in polar regions, offering a significant improvement in latency. The results demonstrate the importance of considering inclination and altitude changes as well as topological variations in satellite constellations when designing efficient routing algorithms for global coverage.

6.1 Future works

Future research could build on the findings of this work by further refining the heuristic and exploring its applicability to other constellation types and network conditions.
For instance, in the introduced heuristic, there is potential for further optimization in constructing routes that traverse polar areas. Rather than using the current approach of dividing the route into 3 segments, where the first segment connects the source to a point in the polar region (polar source), the second connects the polar source to another point in the polar region (polar destination), and the third connects the polar destination to the final destination, a more efficient method could be employed. Specifically, a new criterion could be developed to identify a single optimal point within the polar region. Both the source and destination could

then be directly connected to this single polar point, effectively reducing the route to just 2 segments. The route would simply consist of 2 direct connections: one from the source to the chosen polar point, and the second from the polar point to the destination, reducing the overall route complexity and improving latency.

The issue at the Arctic and Antarctic zones highlights a criticality in the general formalism of routing algorithms. It could be interesting, as a further approach, to use a routing strategy that near the poles does not confine the geometry of possible hops only to the 4 neighbours of each satellite (left, right, previous and successive).

The performance evaluation of the routing algorithms in this thesis has been conducted on a static constellation model. While this approach provides valuable insights, the analysis can be expanded by considering a dynamic constellation, where the positions of satellites and links vary over time. A dynamic model would allow for a more realistic simulation of satellite networks, capturing the temporal fluctuations in satellite connectivity and node availability.

Moreover, expanding the analysis beyond latency, which was the primary performance metric in this study, would enable a deeper understanding of how routing algorithms perform under different operational conditions. Other metrics, such as queue size at each node, traffic load distribution, load balancing efficiency, and link quality, could become more critical depending on the specific application or network context.
For example, in scenarios where algorithms prioritize minimum hop count, once the search space is restricted to a rectangular graph, it would be beneficial to further refine the weights of each edge based on a traffic model that incorporates elements of uncertainty. This approach could allow the routing algorithm to dynamically adjust to fluctuating traffic conditions, prioritizing paths that can better handle congestion or varying traffic loads. By assigning weights to edges that reflect real-time conditions, such as link quality, traffic volume, and node buffer capacity, an algorithm could make more informed routing decisions, enhancing overall network performance.
Incorporating such metrics into the analysis would also enable a more comprehensive study of load balancing across the network. Instead of focusing only on minimizing latency, the routing algorithms could be optimized to distribute traffic more evenly among nodes, preventing bottlenecks in high-traffic areas. Additionally, queue sizes at each node could be monitored to ensure efficient handling of data packets, and link quality could be factored in to prioritize routes with more stable and reliable connections.

In addition to focusing only on the deterministic routing strategies, it would be intriguing to explore machine learning techniques, particularly Reinforcement Learning (Q-learning), for satellite networks. Deterministic methods offer predictable routing but may lack flexibility in dynamic environments, such as those with fluctuating traffic or varying link quality. By implementing Q-learning, network nodes could act as agents that learn optimal routing strategies over time, based on real-time feedback.
A comparative analysis of deterministic versus Q-learning approaches would offer valuable insights. Deterministic methods might excel in static or predictable scenarios, but Reinforcement Learning algorithms would likely offer superior adaptability and load balancing in more complex, real-time conditions. This combination could significantly enhance routing resilience in satellite networks.

Appendix A

Useful MATLAB functions

- Functions id_to_pair and pair_to_id

```
function [o, i]= id_to_pair(Q, id)
%from single ID to pair {o,i}
o = floor((id-1) / Q);
i = mod((id-1), Q);
end
```

```
function id = pair_to_id(Q, o, i)
        %get sat index from pair (o,i)
        id = o * Q + i + 1;
end
```

- Functions get_left and get_right

```
function [o_left,i_left] = get_left(Q, F, P, o,i)
    % get sat on the left of the current one
    if o ~= 0
        o_left = o - 1;
        i_left = i;
    else
        o_left = P - 1;
        i_left = mod(i - F, Q);
    end
end
```

```
function [o_right, i_right] = get_right(Q, F, P, o,i)
    %get sat on the right of the current one
    if o ~= P - 1
        o_right = o + 1;
        i_right = i;
    else
        o_right = 0;
        i_right = mod(i + F, Q);
    end
end
```

- Functions get_successive and get_previous

```
function [o_succ, i_succ] = get_successive(Q, o,i)
%get successive sat indices in the same orbit
o_succ = o;
i_succ = mod(i + 1, Q);
end
```

```
function [o_prev, i_prev] = get_previous(Q, o, i)
%get previous
o_prev = o;
i_prev = mod(i - 1, Q);
end
```

- Function isAscending

```
function tf = isAscending(o_test,i_test,PQ,P,F)
            %compute u_test of sat(o_test,i_test) from walkerDelta
    param
            Q = PQ/P;
            delta_phi = 2 * pi / Q;
            delta_f = F * 2 * pi / PQ;
            u_test = normalize_pi(o_test*delta_f + i_test*delta_phi);
            % Check if the satellite is in ascending node
            tf = (-pi/2 <= u_test) && (u_test <= pi/2);
end
```

- Function plot_path on 3D constellation

```
function plot_path (constellation, path_indices,Q)
    path_indices_o_i = zeros(length(path_indices), 2);
    for idx = 1:length(path_indices)
        [o, i] = id_to_pair(Q, path_indices(idx));
        path_indices_o_i(idx, :) = [o, i];
    end

    colors_flag = diff(path_indices_o_i(:,1));

    %plot first and last sat nodes
    for i=1:numel(path_indices)
      constellation(path_indices(i)).MarkerSize = 12;
      if i == 1 || i == numel(path_indices)
          constellation(path_indices(i)).MarkerColor = [0.8500 0.3250
    0.0980];
      else
          constellation(path_indices(i)).MarkerColor = [0.4660 0.6740
    0.1880];
      end
    end
    %plot links
    for i = 1: numel(path_indices)-1
        gimbaltxSat = gimbal(constellation(path_indices(i)));
        gimbalrxSat = gimbal(constellation(path_indices(i+1)));

        rxSat = receiver(gimbalrxSat);
        txSat = transmitter(gimbaltxSat);
        pointAt(gimbaltxSat,constellation(path_indices(i+1)));
        pointAt(gimbalrxSat,constellation(path_indices(i)));
        %link
        lnk = link(txSat,rxSat);
        set(lnk, 'LineWidth', 4 );
        if colors_flag(i) == 0
            set(lnk, 'LineColor', 'red')
        else
            set(lnk, 'LineColor', 'green')
        end
    end
end
```

Bibliography

[1] Wanshi Chen, Xingqin Lin, Juho Lee, Antti Toskala, Shu Sun, Carla Fabiana Chiasserini, and Lingjia Liu. «5G-Advanced Toward 6G: Past, Present, and Future». In: *IEEE Journal on Selected Areas in Communications* 41.6 (2023), pp. 1592–1619. DOI: 10.1109/JSAC.2023.3274037 (cit. on p. 3).

[2] Shanzhi Chen, Ying-Chang Liang, Shaohui Sun, Shaoli Kang, Wenchi Cheng, and Mugen Peng. «Vision, Requirements, and Technology Trend of 6G: How to Tackle the Challenges of System Coverage, Capacity, User Data-Rate and Movement Speed». In: *IEEE Wireless Communications* 27.2 (2020), pp. 218–228. DOI: 10.1109/MWC.001.1900333 (cit. on p. 3).

[3] *6G: The Next Horizon*. White Paper. Huawei, Jan. 2022. URL: https://www.huawei.com/en/huaweitech/future-technologies/6g-white-paper (cit. on p. 4).

[4] Hejia Luo et al. «Very-Low-Earth-Orbit Satellite Networks for 6G». In: 2022. URL: https://api.semanticscholar.org/CorpusID:263777758 (cit. on p. 4).

[5] Gregory Stock, Juan A. Fraire, and Holger Hermanns. «Distributed On-Demand Routing for LEO Mega-Constellations: A Starlink Case Study». In: *2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*. 2022, pp. 1–8. DOI: 10.1109/ASMS/SPSC55670.2022.9914716 (cit. on pp. 6, 22, 32–35, 37, 38, 49, 53, 57, 84).

[6] Dennis Roddy. *Satellite Communications*. 4th. New York: McGraw-Hill, 2006 (cit. on p. 9).

[7] Juan Fraire. *Space Applications, Physics and Orbits*. Space Networks. 2024 (cit. on pp. 9, 12).

[8] Shadab Mahboob and Lingjia Liu. «Revolutionizing Future Connectivity: A Contemporary Survey on AI-Empowered Satellite-Based Non-Terrestrial Networks in 6G». In: *IEEE Communications Surveys and Tutorials* 26.2

(2024), pp. 1279–1321. DOI: 10.1109/COMST.2023.3347145 (cit. on pp. 12–14).

[9] Gabriel Maiolini Capez. *Doppler effect*. Introduction to Low-Earth Orbit Satellite Link Budgets. 2024 (cit. on p. 16).

[10] Tiejun Wang, J.G. Proakis, E. Masry, and J.R. Zeidler. «Performance degradation of OFDM systems due to Doppler spreading». In: *IEEE Transactions on Wireless Communications* 5.6 (2006), pp. 1422–1432. DOI: 10.1109/TWC.2006.1638663 (cit. on p. 16).

[11] J. G. Walker. «Satellite Constellations». In: *Journal of the British Interplanetary Society* 37.12 (1984), pp. 559–572 (cit. on pp. 16, 17).

[12] Arthur H. Ballard. «Rosette Constellations of Earth Satellites». In: *IEEE Transactions on Aerospace and Electronic Systems* AES-16 (1980), pp. 656–673. URL: https://api.semanticscholar.org/CorpusID:25417933 (cit. on p. 17).

[13] European Space Agency. *Galileo Satellites*. 2024. URL: https://www.esa.int/Applications/Satellite_navigation/Galileo/Galileo_satellites (cit. on p. 18).

[14] The MathWorks, Inc. *SatelliteScenario.WalkerStar*. Accessed: 2024-10-14. 2024. URL: https://it.mathworks.com/help/aerotbx/ug/satellitescenario.walkerstar.html (cit. on p. 18).

[15] Behrouz A. Forouzan. *Data Communications and Networking*. 5th. McGraw-Hill, 2013 (cit. on p. 20).

[16] Qi Xiaogang, Ma Jiulong, Wu Dan, Liu Lifang, and Hu Shaolin. «A Survey of Routing Techniques for Satellite Networks». In: *Journal of Communications and Information Networks* 1.4 (2016), pp. 66–85. DOI: 10.11959/j.issn.2096-1081.2016.058 (cit. on pp. 20, 26).

[17] Quan Chen, Giovanni Giambene, Lei Yang, Chengguang Fan, and Xiaoqian Chen. «Analysis of Inter-Satellite Link Paths for LEO Mega-Constellation Networks». In: *IEEE Transactions on Vehicular Technology* 70.3 (2021), pp. 2743–2755. DOI: 10.1109/TVT.2021.3058126 (cit. on pp. 22, 34).

[18] E.W. Dijkstra. «A Note on Two Problems in Connexion with Graphs». In: *Numerische Mathematik* 1.1 (1959), pp. 269–271 (cit. on pp. 22, 41).

[19] Zhaolong Ding, Huijie Liu, Feng Tian, Zijian Yang, and Nan Wang. *Fast-Convergence Reinforcement Learning for Routing in LEO Satellite Networks*. Apr. 2023. DOI: 10.20944/preprints202304.0656.v1 (cit. on pp. 22, 28).

[20] J. Virgili Llop, P.C.E. Roberts, Zhou Hao, L. Ramio Tomas, and V. Beauplet. «Very Low Earth Orbit mission concepts for Earth Observation: Benefits and challenges.» English. In: *Reinventing Space Conference*. Reinventing Space Conference ; Conference date: 18-11-2014 Through 20-11-2014. Nov. 2014 (cit. on p. 24).

[21] Lucy Berthoud, Russell Hills, Andrew Bacon, Michael Havouzaris-Waller, Kieran Hayward, Jean Gayrard, Fabrice Arnal, and Laurent Combelles. «Are Very Low Earth Orbit (VLEO) satellites a solution for tomorrow's telecommunication needs?» In: *CEAS Space Journal* 14 (June 2022). DOI: 10.1007/s12567-022-00437-0 (cit. on p. 25).

[22] Nicholas Crisp et al. *The Benefits of Very Low Earth Orbit for Earth Observation Missions*. Dec. 2019. DOI: 10.48550/arXiv.2007.07699 (cit. on p. 25).

[23] N. H. Crisp et al. «System modelling of very low Earth orbit satellites for Earth observation». English. In: *Acta Astronautica* 187 (July 2021), pp. 475–491. ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2021.07.004 (cit. on p. 25).

[24] «Very Low Earth Orbit Constellations for Earth Observation». English. In: *73rd International Astronautical Congress*. 73rd International Astronautical Congress, IAC 2022 ; Conference date: 18-09-2022 Through 22-09-2022. Sept. 2022 (cit. on p. 26).

[25] Chengcheng Li, Yasheng Zhang, Zixuan Cui, Yi Zhang, Jiamin Liu, Zilong Yu, and Peiying Zhang. «An Overview Of Low Earth Orbit Satellite Routing Algorithms». In: *2023 International Wireless Communications and Mobile Computing (IWCMC)*. 2023, pp. 866–870. DOI: 10.1109/IWCMC58020.2023.10182766 (cit. on p. 26).

[26] Zhenzhen Han, Chuan Xu, Guofeng Zhao, Shanshan Wang, Kefei Cheng, and Shui Yu. «Time-Varying Topology Model for Dynamic Routing in LEO Satellite Constellation Networks». In: *IEEE Transactions on Vehicular Technology* 72.3 (2023), pp. 3440–3454. DOI: 10.1109/TVT.2022.3217952 (cit. on p. 27).

[27] Hong Seong Chang, Byoung Wan Kim, Chang Gun Lee, Yanghee Choi, Sang Lyul Min, Hyun Suk Yang, and Chong Sang Kim. «Topological design and routing for low-Earth orbit satellite networks». In: *Proceedings of GLOBECOM '95*. Vol. 1. 1995, 529–535 vol.1. DOI: 10.1109/GLOCOM.1995.501983 (cit. on p. 27).

[28] Hong Seong Chang, Byoung Wan Kim, Chang Gun Lee, Sang Lyul Min, Yanghee Choi, Hyun Suk Yang, Doug Nyun Kim, and Chong Sang Kim. «Performance comparison of static routing and dynamic routing in low-Earth orbit satellite networks». In: *Proceedings of Vehicular Technology Conference - VTC*. Vol. 2. 1996, 1240–1243 vol.2. DOI: 10.1109/VETEC.1996.501510 (cit. on p. 27).

[29] Peiliang Zuo, Chen Wang, Ze Yao, Shaolong Hou, and Hua Jiang. «An Intelligent Routing Algorithm for LEO Satellites Based on Deep Reinforcement Learning». In: *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*. 2021, pp. 1–5. DOI: 10.1109/VTC2021-Fall52928.2021.9625325 (cit. on p. 29).

[30] Xiaoting Wang, Zhiqi Dai, and Zhao Xu. «LEO Satellite Network Routing Algorithm Based on Reinforcement Learning». In: *2021 IEEE 4th International Conference on Electronics Technology (ICET)*. 2021, pp. 1105–1109. DOI: 10.1109/ICET51757.2021.9451072 (cit. on p. 29).

[31] National Geospatial-Intelligence Agency. *Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems*. Tech. rep. TR8350.2. National Geospatial-Intelligence Agency, 1991. URL: https://apps.dtic.mil/sti/pdfs/ADA280358.pdf (cit. on p. 32).

[32] SpaceX Space Exploration Holdings LLC. *SpaceX non-geostationary satellite system: Attachment A, FCC IBFS SAT-MOD-20190830-00087*. Aug. 2019. URL: https://fcc.report/IBFS/SAT-MOD-20190830-00087/1877671 (cit. on pp. 50, 63, 67).

[33] SpaceX Space Exploration Holdings LLC. *SpaceX Non-Geostationary Satellite System: Attachment A, FCC IBFS SAT-AMD-20210818-00105*. Aug. 2021. URL: https://fcc.report/IBFS/SAT-AMD-20210818-00105/12943362 (cit. on pp. 50, 63, 67).