Advanced Routing Strategies for LEO and VLEO Constellations: Ensuring Polar Coverage

Advanced Routing Strategies for LEO and VLEO Constellations: Ensuring Polar Coverage

Camilla Ottaviani*, Alessandro Compagnoni*, Juan A. Fraire†‡, Giacomo Verardo¶, Gabriel Maiolini Capez*, Daniel Gaetano Riviello§, Gregory Stock‖, Carla Fabiana Chiasserini*, Roberto Garello*
*Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Turin, Italy
†Inria, INSA Lyon, CITI, UR3720, 69621 Villeurbanne, France
‡CONICET – Universidad Nacional de Córdoba, Córdoba, Argentina
§CNR-IEIIT, National Research Council of Italy, 10129 Turin, Italy
¶KTH Royal Institute of Technology, Stockholm, Sweden
‖Saarland University, Saarland Informatics Campus, 66123 Saarbrücken, Germany

***Abstract*—The proliferation of mega-constellations comprising thousands of satellites has amplified the need for efficient routing solutions in space networks to minimize latency and ensure robust global connectivity. This work evaluates the performance of multiple decentralized routing algorithms within Walker-Delta constellations, comparing their effectiveness in Low Earth Orbit (LEO) and Very Low Earth Orbit (VLEO) configurations. Our findings reveal that constellation geometry, such as the number of satellites and orbital planes, shapes routing strategies more than orbital altitude, with VLEO demanding larger fleets to maintain seamless coverage. Furthermore, we propose *PolarDisCo*, a novel routing algorithm specifically designed to provide coverage of polar regions—a geographical area where, due to the unique coverage and topology challenge it poses, existing decentralized routing algorithms (e.g., *DisCoRoute*) exhibit a significant performance drop. Experimental results demonstrate that *PolarDisCo* reduces end-to-end delay on the polar area by 7.9 ms with respect to *DisCoRoute*, thus providing a 14.8% improvement. These findings underscore the potential of tailored routing strategies to optimize communication in next-generation satellite networks involving LEO and VLEO shells.**

***Index Terms*—LEO and VLEO Satellite Networks, Routing, Performance Evaluation**

I. Introduction

The rapid advancements in aerospace technologies and the significant reduction in satellite production costs, driven by investments from private space companies such as OneWeb and Starlink, have boosted the deployment of mega-constellations in LEO and VLEO. These constellations, consisting of thousands of satellites, aim to provide Non-Terrestrial Network (NTN) services with global connectivity, enhanced reliability, and low-latency communication [1], [2].

LEO constellations operate at altitudes between 500 and 2000 km, whereas VLEO between 300 and 500 km. The lower altitude of VLEO satellites results in stronger signal strength, reduced propagation delays, and enhanced resolution for Earth observation [3], [4]. However, these benefits come at the cost of increased atmospheric drag, shorter satellite lifespan, frequent orbital adjustments, and increased fleet size to ensure continuous coverage [5], [6]. It follows that ensuring global satellite connectivity is still an open problem and, although widely studied in the literature for LEO [7], routing of data traffic in VLEO constellations [8] still poses challenges. First, despite the frequent topology changes caused by satellite motion, it is crucial to guarantee reliable and timely data delivery between satellites and between satellites and ground stations. Secondly, it is essential to offload data from isolated regions, such as the poles, with reduced ground station coverage. This has become even more important because of the strategic relevance of polar areas for environmental monitoring and recent geopolitical concerns. Third, in large fleets—e.g., in VLEO constellations—it is critical to envision *decentralized* routing schemes, which scale well with the number of satellites, opposite to *centralized* approaches that instead struggle with the signaling and computational overhead required to manage frequent topology changes from a central ground center.

In this paper, we address the above challenges by focusing on onboard routing over inter-satellite links (ISLs) in LEO and VLEO constellations. We first present a thorough analysis of state-of-the-art decentralized routing schemes on realistic LEO and VLEO shells from Starlink. Then, we introduce *PolarDisCo*, a heuristic routing algorithm specifically designed to ensure connectivity in the polar regions, where high satellite density and rapid topology changes at higher latitudes make it challenging to maintain ISLs. We remark that conventional routing algorithms struggle with the problem's complexity and fail to provide reliable data routing in the polar regions.

Routing research in VLEO is scarce, with [9] being the only study proposing an RL-based approach for optical ISLs that is centrally trained on the ground. To the best of the authors' knowledge, our work is the first to tackle processing-efficient onboard decentralized routing in VLEO, addressing the unique challenges of polar regions. The main contributions of our work are, therefore, as follows:

1) **Sensitivity Analysis of Routing Algorithms:** After introducing key modeling concepts (Section II), we present a comparative analysis of several routing algorithms for Walker-Delta constellations, examining the impact of altitude in LEO and VLEO, inclination, and processing delays (Section III).
2) **Case Studies in LEO and VLEO Shells:** We investigate the performance of specific LEO and VLEO configura-

tions extracted from Starlink configurations, identifying the key factors that affect routing efficiency (Section IV).
3) **Routing Heuristic for the Polar Regions:** We propose the *PolarDisCo* algorithm, which overcomes the limitations of the state-of-the-art *DisCoRoute* scheme [10], leveraging unique polar topology features to improve routing performance (Section V).

II. System Model

In this section, we first introduce the orbital configuration we consider, namely the Walker-Delta constellations. Then, we briefly describe the decentralized routing algorithms we analyze and the tools we use to perform our analysis.

A. Walker-Delta Constellations

In a Walker-Delta constellation [11], orbits follow a flower-like pattern and are evenly distributed along the Equator (that is why it is also referred to as the *Ballard-Rosette* configuration [12]). Constellations are expressed through the notation $\alpha: T/P/F$, where α is the inclination of orbital planes with respect to the equatorial plane, T is the total number of satellites, P is the number of planes, and F is the *phasing factor*. Orbits are circular; they all have the same inclination and altitude from the surface of the Earth. Q satellites are evenly distributed within each orbit, therefore T can also be referred as the product $P \cdot Q$. The parameter F defines the spacing between satellites in adjacent planes, specifying how much a satellite leads or lags its counterpart in the next plane. When F is zero, all satellites are aligned with their neighbors in adjacent planes. A more detailed explanation can be found in [10].

Each satellite is uniquely identified by the pair of indices (o, i) where $o \in \{0, \ldots, P-1\}$ refers to the orbital plane index and $i \in \{0, \ldots, Q-1\}$ refers to the i-th satellite within that plane. Alternatively, we will refer to a satellite using a single ID ranging from 1 to T. Each satellite has four neighbors: two on the same orbit (one ahead and one behind) and two on adjacent orbits (one on the left and one on the right). The previous neighbor within the same orbit is given by $(o, (i-1) \bmod Q)$ while the next neighbor is $(o, (i+1) \bmod Q)$. For adjacent orbits, the satellite on the left if $o \neq 0$ is $(o-1, i)$ and $(P-1, (i-F) \bmod Q)$ otherwise. The right neighbor is $(o+1, i)$ if $o \neq P-1$ and $(0, (i+F) \bmod Q)$ otherwise.

In the literature, when a node connects to its left or right neighbor, it does so through an *inter-plane link* or *horizontal hop*. If the next node is a satellite within the same orbital plane, the hop is *vertical* and called *intra-plane link*.

B. Routing Algorithms

This section introduces the existing decentralized routing algorithms that we consider in our sensitivity analysis (a summary is also presented in Table I). While presenting these strategies, we assume for simplicity a fixed pair of source and destination satellites.

Some of the strategies that we investigate take as input the results of the Minimum Hop Count (*MinHopCount*) [13], a mathematical formulation of the minimum number of ISL hops required to connect two satellites. *MinHopCount*-based algorithms follow four possible directions to compute a route between source and destination, without going backward: North-West (NW), North-East (NE), South-West (SW), or South-East (SE). For example, NE means that each satellite can forward packets only to its successor satellite on its plane or to its right neighbor. *MinHopCount* computes the number of hops required by each of the four directions, and it returns the overall minimum sum of vertical and horizontal hops, along with the corresponding best direction.

Routing algorithms can use the minimum hop count to reduce their search space to a spherical, rectangular grid with the source and destination on opposite vertices. All possible paths from source to destination have the same number of hops on that grid, which is the minimum one (cf. Manhattan grid). The resulting path may not be associated with the shortest distance/propagation delay. Still, it always has the minimum number of hops, often the preferred metric, as each hop adds some additional delay for on-board processing and queuing. In the following, we present the list of considered algorithms.

a) Dijkstra: Dijkstra [14] is a widely used routing algorithm that guarantees the optimal, shortest-path route selection between source and destination at convergence. In this work, we consider:

- *Dijkstra-Distance* focuses on finding the shortest path in terms of shortest distance, i.e., minimizing the overall route propagation delay [10].
- *Dijkstra-Latency* is an alternative that also considers the processing time given by intermediate nodes of the path.

While Dijkstra-based solutions do not use *MinHopCount*, the algorithms below exploit it to reduce the search space:

b) Trivial: The *Trivial* routing algorithm, once the minimum number of vertical and horizontal hops and their directions to follow are known, completes all the necessary horizontal hops first before proceeding with all the vertical ones. Alternatively, the algorithm performs all the vertical hops first and then all the horizontal ones.

c) Flip-Coin: It follows a similar strategy but flips a coin at each node to randomly decide which of the two directions the packet should continue. When all hops of one type have been completed, the route is filled with the remaining hops of the other type.

d) DisCoRoute [10]: It uses a heuristic to cleverly choose the sequence of hops according to two geometric properties of Walker-Delta topologies: (*i*) all intra-plane hops have the same length, which is constant over time, and (*ii*) inter-plane hops tend to be shorter near the poles and most extended over the Equator. Therefore, the strategy of *DisCoRoute* is to distribute all horizontal hops so that they occur as close to the poles as possible, i.e., at high latitudes (in terms of absolute values).

C. Tools

We conducted experiments to collect and evaluate results from the constellation layouts and the routing algorithms implemented in MATLAB R2024b. The studies involved

Table I: Overview of the examined routing algorithms

Algorithm Name	Objective
Dijkstra-Distance	minimize route length / propagation latency
Dijkstra-Latency	minimize processing/total latency
Trivial	minimize number of hops only
Flip-Coin	minimize number of hops only
DisCoRoute	minimize hops; use heuristic to min. distance
PolarDisCo	*DisCoRoute* + improved polar performance

building satellite scenarios using the default orbit propagator of the Aerospace Toolbox for Satellite Mission Analysis and the Satellite Communication Toolbox. We utilized a strategy based on proximity within the Walker-Delta pattern to select which Inter-Satellite Links (ISLs) to establish with neighboring satellites. The set-up included an HP Z2 Tower G9 Workstation, with an Intel Core i9-13900 processor and 32 GB of RAM.

III. Sensitivity Analysis Across Altitude, Inclination, and Packet Processing Time

Most of the algorithms analyzed in this study have previously been compared in [10] on the LEO shell $53°: 1584/72/39$ at 550 km of the Starlink constellation. The key findings reported in this paper are that, on this particular shell, *DisCoRoute* provides running times two orders of magnitude lower than *Dijkstra-Distance* while generating competitive routes in terms of propagation delay (only 2% longer in the worst case).

In this section, we use another constellation (namely $96.9°: 2000/40/21$) as a generic baseline to examine how variations in inclination and altitude influence the end-to-end routing delay over 5,000 randomly selected satellite pairs. The altitude range varies from 360 km (a typical VLEO altitude) to 2000 km (covering most LEO ranges), while the inclinations vary from 50° to 130°. Furthermore, we analyze the impact of processing delay (encompassing queuing, onboard computation, and transmission delays) across a range of 0 to 100 ms on every intermediate route hop.

A. Latency vs. Altitude, Inclination, and Processing Time

When we increase the altitude of the constellation, keeping the inclination at 96.9° and the processing time at zero (thus, *Dijkstra-Distance* and *Dijkstra-Latency* are equivalent), the average delay grows linearly for all the routing algorithms, as shown in Figure 1a, with *Trivial* and *Flip-Coin* performing poorly with respect to *DisCoRoute*, *PolarDisCo*, and *Dijkstra-Distance*. Instead, by increasing only the inclination, the latency is significantly reduced. In particular, *PolarDisCo* and *Dijkstra-Distance* perform better than the others, as highlighted in Figure 1b. In general, *DisCoRoute* and all *MinHopCount*-based algorithms work well when hops have similar distances. In this scenario, minimizing the number of hops is almost equivalent to computing the shortest distance.

The analysis presented in Figure 1c takes *Dijkstra-Latency* as the baseline for performance comparison. This algorithm aims to minimize the overall end-to-end delay. High processing time values imply high costs for each node; hence, *Dijkstra-Latency* will tend to select paths with the minimum number of hops. On the other hand, *Dijkstra-Distance* still selects the shortest path in terms of distance, which may not be the best choice since traffic may traverse more (highly costly) nodes. Indeed, for high processing times, *MinHopCount*-based algorithms, especially *DisCoRoute*, perform better than *Dijkstra-Distance*.

B. Latency vs. Altitude: Theoretical Model

One of our main objectives is to understand how different routing strategies behave in a VLEO scenario where the orbit radius is smaller than in LEO constellations. However, as already observed from Figure 1a, the average end-to-end delay provided by different algorithms increases linearly with orbital altitude, all with the same slope. Here, we aim to demonstrate that different routing strategies are not affected by orbital altitude and, as an immediate consequence, end-to-end delays increase linearly with it. Hereafter, orbital radius and altitude will be used interchangeably, as they differ only by the Earth's radius, which is constant.

Given a network with T satellites, we denote the set of distances with $\{d_{i,j}\}_{i,j=1,\dots,T,\, i\neq j}$, with $d_{i,j}$ representing the distance between satellites i and j. A path connecting a source and a destination node $s,t \in \{1,\dots,T\}$ is described as a set of connected pairs from s to t, i.e., $p \triangleq \{(s,i),(i,k),\dots,(j,u),(u,t)\}$.

Definition 1 (Distance-based routing strategy)*:* We can think of a Distance-based Routing Strategy (DBRS) as an algorithm that, among some paths between two nodes, selects the one with the shortest distance. More formally, given a network with T nodes, and any source and destination pair $s,t \in \{1,\dots,T\}$, a DBRS selects the path between s and t according to

$$p^* = \arg\min_{p\in\mathcal{P}_{s,t}} \sum_{i\in p} d_{p(i)} \tag{1}$$

where $\mathcal{P}_{s,t}$ is a set of paths between s and t, and $p(i)$ denotes the i-th pair on path p. Note that if $\mathcal{P}_{s,t}$ contains all the possible paths between s and t, then the DBRS provides the optimal solution.

Theorem 1*:* A DBRS does not depend on the specific values of the distances $\{d_{i,j}\}_{i,j=1,\dots,T,\, i\neq j}$, but only on their ratios.

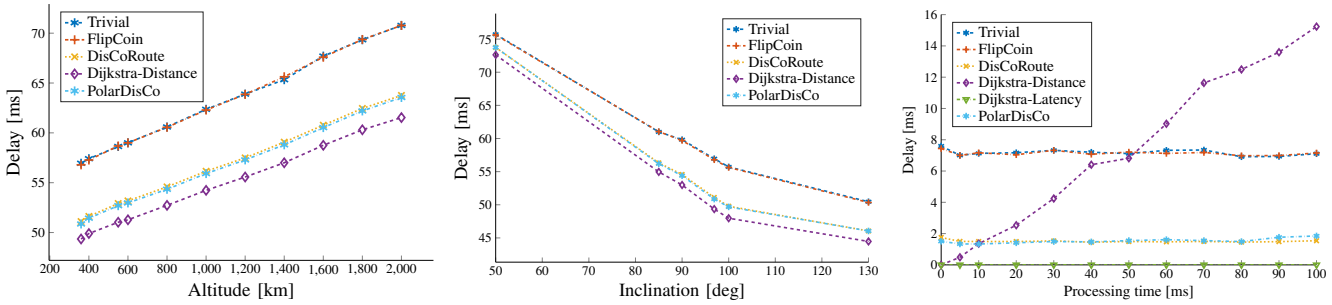
Proof. Given the solution p^* of the DBRS over the set of distances $\{d_{i,j}\}_{i,j=1,\dots,T,\, i\neq j}$, that is

$$p^* = \arg\min_{p\in\mathcal{P}_{s,t}} \sum_{i\in p} d_{p(i)}$$

we multiply each distance by a constant $a > 0$, preserving their mutual ratios. Applying the DBRS on $\{a \cdot d_{i,j}\}_{i,j=1,\dots,T,\, i\neq j}$ yields the same path, as the scaling factor a does not alter the minimization argument. □

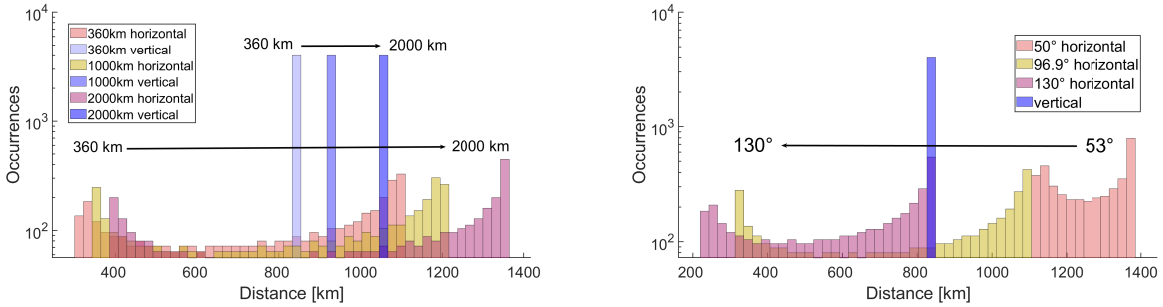
Theorem 2*:* In Walker-Delta constellations, distance-based routing strategies do not depend on the orbit radius.

Proof. In a Walker-Delta constellation, the orbits are circular and share the same center; thus, the satellites are positioned on the surface of a sphere with radius r. Given any two points on this sphere, they lie on at least one great circle (exactly one if they are not antipodal, ∞ if they are antipodal). Therefore,



(a) Varying h (b) Varying α (c) Varying PT (w.r.t. *Dijkstra-Latency*)

Figure 1: Latency vs. Altitude, Inclination and Processing Time (PT) for configuration 2000/40/21, $\alpha = 96.9°$, $h = 360\,\text{km}$



(a) $\alpha = 96.9°$, varying h (b) $h = 360\,\text{km}$, varying α

Figure 2: Number of distance occurrences for configuration 2000/40/21

their distance can be expressed as the chord length connecting them along the great circle. Let us now consider the distance between two satellites j and k. By the chord properties, we can express it as:

$$d_{j,k} = 2r \sin\left(\frac{\theta_{j,k}}{2}\right)$$

where $\theta_{j,k}$ is the central angle formed between satellite j, the center of the great circle, and satellite k.

Thus, we can express the set of distances between pairs of satellites as $\{2r \sin(\theta_{i,j}/2)\}_{i,j=1,\ldots,T,\, i\neq j}$. Dividing each distance by the constant r, we get: $\{2 \sin(\theta_{i,j}/2)\}_{i,j=1,\ldots,T,\, i\neq j}$ and applying Theorem 1 proves the claim. □

Corollary 1: The end-to-end delay given by the solutions of DBRS increases linearly with r in Walker-Delta constellations.

Proof. Without loss of generality, assume that the path provided by the solution of the DBRS is

$$\{(1,2),(2,3),\ldots,(K-1,K)\}$$

for some positive integer K. Then its length is given by

$$\sum_{i=1}^{K-1} d_{i,i+1} = \sum_{i=1}^{K-1} 2r \sin\left(\frac{\theta_{i,i+1}}{2}\right) = 2r \sum_{i=1}^{K-1} \sin\left(\frac{\theta_{i,i+1}}{2}\right).$$

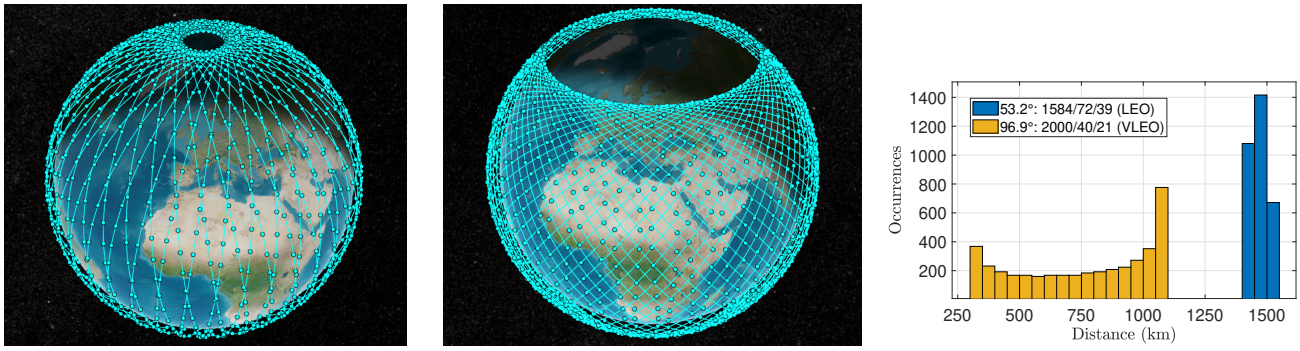
Hence, the path length clearly grows linearly with r, independently of the adopted distance-based routing strategy, and so does the propagation delay. □

Figure 2a illustrates the histogram of the distance for satellites in the baseline constellation at various altitudes. As just anticipated, the histogram maintains a consistent shape across different altitudes, while shifting linearly as the altitude changes.

IV. Case Studies in LEO and VLEO Constellation Shells

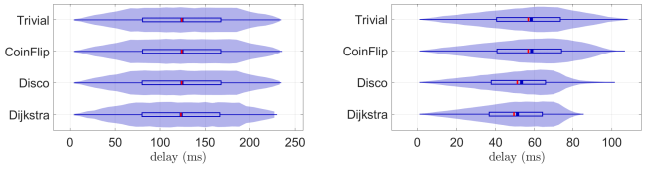
This section assesses the end-to-end delay distribution of the evaluated routing solutions in typical LEO and VLEO configurations. To this end, we focus on two specific orbital shells obtained from Starlink public data: 53.2°: 1584/72/39 at an altitude of 540 km [15], and 96.9°: 2000/40/21 at an altitude of 360 km [16]. They are illustrated in Figure 3a and Figure 3b. For simplicity, we will henceforth refer to these configurations as "*LEO*" and "*VLEO*", respectively.

When considering routing, we should highlight that the geometry of a specific constellation can either accentuate or diminish the performance gap between different routing strategies. Figure 3c underlines that the histogram of inter-plane distances in *VLEO* has a larger support than that in *LEO*. In fact, *LEO* features more orbital planes than *VLEO*, resulting in neighboring satellites in adjacent orbits maintaining relatively stable distances from one another as they travel between the equatorial and polar regions. Hence, this reduces the range of possible distances. In contrast, *VLEO*, with fewer orbital planes, shows more pronounced variations in the distance between satellites in adjacent orbits as they transition from the Equator to the poles. Therefore, as a rule of thumb, there is a more



(a) 3D plot of 96.9° : 2000/40/21 (*VLEO*) (b) 3D plot of 53.2° : 1584/72/39 (*LEO*) (c) Inter-plane distance histogram

Figure 3: 3D plots of *LEO* and *VLEO* constellations with number of inter-plane distance occurrences comparison



(a) *LEO* (b) *VLEO*

Figure 4: End-to-end delay distribution for *LEO* and *VLEO*

significant performance gap between "good" and "bad" routing strategies in constellations with fewer orbital planes.

A similar observation applies to the inclination parameter: higher inclinations cause neighboring satellites in adjacent orbits to be very close to each other near the poles but much farther apart at the Equator. Since *VLEO* has both a higher inclination and fewer orbital planes, we expect that an optimized routing strategy could achieve considerable performance gains in this scenario. Figure 2b shows the histogram of distances for satellites in the *VLEO* constellation, where we tested different inclinations. As the inclination increases, the magnitude of the intra-plane links remains unchanged while the inter-plane distances decrease and, as expected, the range of occurrences increases.

The previous statements are further motivated in Figure 4, which present the end-to-end delay distributions produced by different routing algorithms obtained by selecting 150,000 random source-destination satellite pairs. For this analysis, we assumed a processing time equal to zero.

In Figure 4a, representing the *LEO* case, the algorithms' performance difference is minimal, as the distances between sources and destinations exhibit little variability across different paths. Conversely, Figure 4b reveals a marked performance gap in the *VLEO* scenario, where the *Dijkstra* and *DisCoRoute* algorithms outperform the others. This result highlights the impact of constellation geometry on routing performance, showing that in specific satellite scenarios, effective routing strategies lead to better network performance and efficiency. Comparing the two figures, it is clear that delays in *VLEO* are smaller, with an average between 50 and 60 ms among different algorithms, compared to approximately 125 ms for *LEO*.

V. PolarDisCo: A Routing Heuristic for Polar Coverage

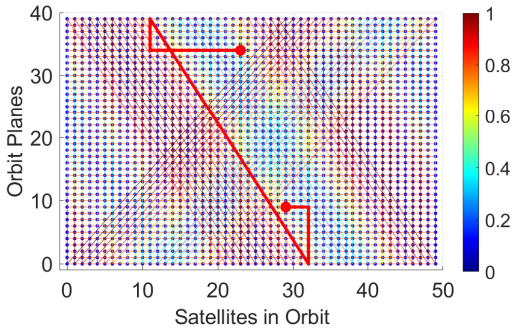
In this section, we first analyze the topology of the polar regions and identify low-cost areas that may benefit from different types of routing. Then, we provide the conditions of applications for our *PolarDisCo* algorithm, and finally, we describe *PolarDisCo* and evaluate its performance against *DisCoRoute*. Table II summarizes our system notation.

A. Analysis in Polar Regions

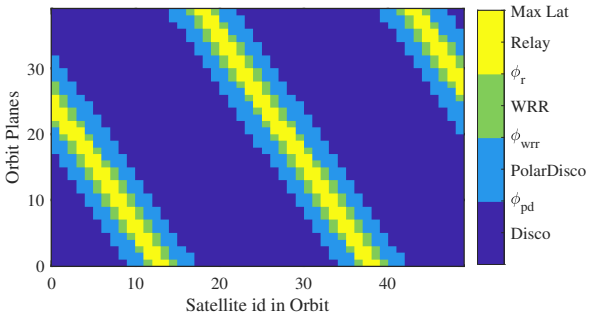
Figure 5a shows that when *DisCoRoute* is employed in the 96.9° : 2000/40/21 layout, the potential advantages of shorter link distances in the polar regions are not realized as expected. In the case of *DisCoRoute*, the chosen route avoids the light blue area of the grid, suggesting that these regions, which could provide more efficient connections due to shorter link lengths, are not being utilized.

In contrast, routes passing through polar satellites do not increase performance when the source and destination are not in the polar region themselves. On the other hand, we have identified two polar sub-regions near the poles where the optimal route between two satellites in those regions is qualitatively different. We refer to them as *PolarDisCo* and *Weighted Round Robin (WRR)-Polar* zone, as shown in Figure 5b.

In *PolarDisCo*, satellites have latitude ϕ, which satisfies the inequality $\phi_{pd} \leq |\phi| \leq \phi_{wrr}$, i.e., they are close to the pole but not exactly on the pole. In such an area, the path is split into three distinct paths using two relay nodes in the pole. The *WRR-Polar* area is the closest to the poles, with latitude above ϕ_{wrr} in absolute value. At such high latitudes, performing hops in a straight line between source and destination in the same zone is sufficient to achieve a small end-to-end delay. Hence, the *WRR* and *PolarDisCo* zones require different routing algorithms to minimize the end-to-end delay. The color bar in Figure 5b shows the latitude thresholds and corresponding zones for applying *PolarDisCo* ($\phi_{pd} = 60°$) and WRR routing ($\phi_{wrr} = 75°$). The *Relay* zone ($\phi_r = 80°$) indicates the region where satellites are used as a relay for the *PolarDisCo* zone. The threshold latitudes were chosen empirically by evaluating multiple combinations, as explained in Section V-C.



(a) *DisCoRoute* path from *src* $(34, 23)$ to *dst* $(9, 29)$



(b) Different zones in the polar area with our proposed algorithm.

Figure 5: 2D grid of 96.9°: 2000/40/21 with distance-based colored links with routing paths

B. PolarDisCo Algorithm

The benefits of employing our heuristic reside in the availability of low-cost links in the polar region. Hence, to reap the advantages of such zones, we devise a set of conditions for applying *PolarDisCo*. In particular, we employ three conditions on either latitude of source and destination satellites (Figure 5b) and a number of hops between them:

$$\phi_{wrr} \geq \phi_{src} \geq \phi_{pd} \quad \text{and} \quad \phi_{wrr} \geq \phi_{dst} \geq \phi_{pd} \tag{2}$$

$$\phi_{src} \geq \phi_{wrr} \quad \text{and} \quad \phi_{dst} \geq \phi_{wrr} \tag{3}$$

$$n_h + n_v \geq N_{pd} \tag{4}$$

where ϕ_{pd} and ϕ_{wrr} are latitude thresholds to apply PolarDisCo and WRR respectively (in the North Pole), while N_{pd} is a lower bound on the number of hops. The latitude conditions are equivalently applied to the South Pole. Equation 2 allows performing our heuristic only when both source and destination are close to the poles (the edge zone in Figure 5b). Based on Equation 3, WRR is instead applied on a deeper zone close to the poles (an intermediate region in Figure 5b), where a straight route between the satellites allows gaining the most from the link in the polar region. We employ instead Equation 4 to avoid costly back-and-forth routes from the polar region when a limited gain could be obtained due to the negligible number of hops between source and destination satellite.

Algorithm 2 describes the routing algorithm in the *PolarDisCo* zone, where we consider three sub-routes in the path between source and destination: (i) $Path_{s2r}$ between source and source relay node, (ii) $Path_{r2r}$ between the two relay nodes in the pole region, (iii) $Path_{r2d}$ between destination relay node and destination. The relay nodes are satellites taken from the *Relay* region and used as intermediate points in the route. The path between the source and the source relay is established with the *Trivial* algorithm, with all the vertical hops followed by the horizontal ones (similarly for the destination).

For the relay-to-relay path, a combination of $k \times k$ relay nodes are considered, where the k closest satellites to the source and destination are evaluated as candidates. Equation 5 measures the gain in employing a path between the relay nodes, as it considers both numbers of vertical and horizontal hops, where horizontal hops are scaled by a function of the latitude of the satellites in the path. Hence, the closer the path is in the polar region, the lower the metric:

$$m_{r2r}(P) = \min_{d \in D} \left(n_v + n_h \frac{1}{|P|} \sum_{i \in P} \left(1 - \left| \frac{\phi_i}{90} \right| \right) \right) \tag{5}$$

where $D = \{SE, NE, NW, SW\}$ is the set of four directions between source and destination, while P is the path connecting the two, and ϕ_i is the latitude of the i-th satellite in P in degrees. By weighting the number of horizontal hops as a function of their latitude, we can reduce the metric corresponding to a route with hops close to the poles. Such a route is indeed beneficial since horizontal hops have a negligible impact on the end-to-end delay when close to the poles.

Algorithm 1 Weighted Round Robin Routing

Require: s_{src}: source satellite, n_h, n_v: number of horizontal and vertical hops, d: direction
Ensure: *Path*: route from source to destination
1: $s_{curr} \leftarrow s_{src}$ {Current satellite is the source satellite}
2: $count_h \leftarrow 0$ {No horizontal hops taken at startup}
3: **for** $i = 1$ to $n_h + n_v$ **do**
4: **if** $\frac{count_h}{i-1} < \frac{n_h}{n_h+n_v}$ **then**
5: $count_h \leftarrow count_h + 1$
6: $s_{curr} \leftarrow next_{h,d}(s_{curr})$ {Next H hop in direction d}
7: **else**
8: $s_{curr} \leftarrow next_{v,d}(s_{curr})$ {Next V hop in direction d}
9: **end if**
10: $Path \leftarrow [Path, s_{curr}]$
11: **end for**
12: **return** *Path*

A WRR path is computed according to Algorithm 1 between all the possible k-by-k pairs of relay satellites. The main goal of the WRR algorithm is to generate a route that only slightly deviates during the path. To do that, the number of horizontal hops taken is recorded at each selection of the next satellite node (line 5). The algorithm aims to maintain the ratio between this number and the total number of taken hops to $\frac{n_h}{n_h+n_v}$, thus not deviating from the expected ratio between horizontal and total hops. The obtained path thus stands in a straight line between the two nodes, making the algorithm suitable when

we aim to remain in the polar region (as in the relay-to-relay sub-route).

Equation 5 is used to evaluate the gain of using each possible relay pair. Moreover, the number of hops between source, destination, and relay nodes is also considered. The solution minimizing the metric from line 12 in Algorithm 2 is selected in line 15. The three subroutes are then concatenated, thus providing an efficient path that transverses the polar region with a low overall propagation time.

Table II: System Notation

Symbol	Description	Unit
h	Constellation altitude	m
α	Constellation inclination	Deg
s_i	Satellite i	-
ϕ_i	Latitude of s_i	Deg
n_h	Number of horizontal hops	-
n_v	Number of vertical hops	-
ϕ_{src}	Source node latitude	Deg
ϕ_{dst}	Destination node latitude	Deg
ϕ_{pd}	Min. latitude to apply PolarDisCo	Deg
ϕ_{wrr}	Min. latitude to apply WRR	Deg
ϕ_r	Min. latitude to identify relay nodes	Deg
N_{pd}	Min. number of hops to apply PolarDisCo	-
k	Number of candidate relay nodes	-
P	Path between nodes	-
d	Direction of path	-
D	Set of possible directions	-
$d_{hop}(s_i, s_j)$	Min. number of hops between s_i and s_j	-

Algorithm 2 PolarDisCo Routing

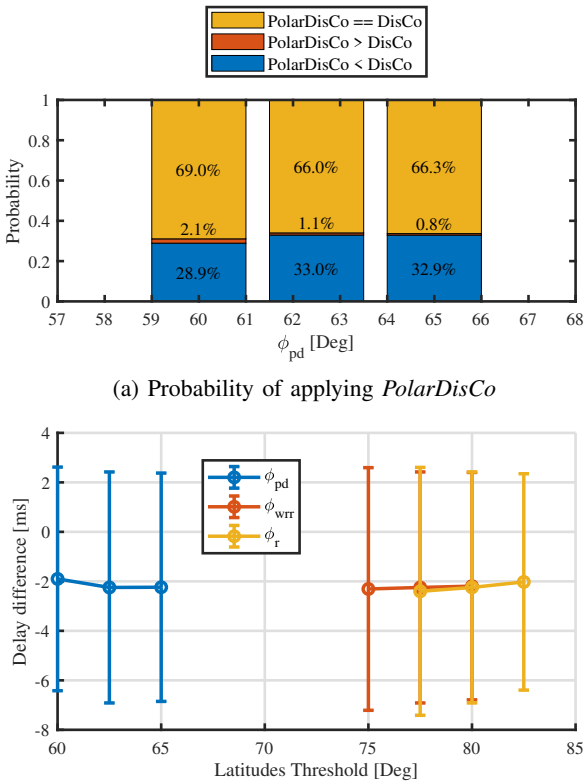
Require: s_{src}, s_{dst}: source and destination satellites, $\{\phi_i\}$: satellite latitudes, ϕ_r: latitude threshold for relay satellites, k: number of candidate relay satellites

Ensure: *Path*: route from source to destination

1: $S_r = \{s_i \text{ s.t. } |\phi_i| > \phi_r\}$ {Set of relay satellites}
2: $S_{r,src} = \arg\min_{\substack{s_i \in S_r \\ |S_{r,src}|=k}} d_{hop}(s_{src}, s_i)$
3: $S_{r,dst} = \arg\min_{\substack{s_i \in S_r \\ |S_{r,dst}|=k}} d_{hop}(s_{dst}, s_i)$
4: $M \in \mathbb{R}^{k \times k}$ initialized to 0
5: {Iterate over all the relay pairs}
6: **for** $s_{r,src} \in S_{r,src}$ **do**
7: **for** $s_{r,dst} \in S_{r,dst}$ **do**
8: $m_{s2r} \leftarrow d_{hop}(s_{src}, s_{r,src})$
9: $m_{r2d} \leftarrow d_{hop}(s_{r,dst}, s_{dst})$
10: $P \leftarrow$ Algorithm 1 between $s_{r,src}$ and $s_{r,dst}$
11: $m_{r2r} \leftarrow$ Equation 5 over P
12: $M\left(s_{r,src}, s_{r,dst}\right) \leftarrow m_{s2r} + m_{r2r} + m_{r2d}$
13: **end for**
14: **end for**
15: $s^*_{r,src}, s^*_{r,dst} \leftarrow \arg\min_{s \in M}$ {Select best relay pair}
16: $Path_{s2r} \leftarrow \text{trivial}(s_{src}, s^*_{r,src})$ {Source to relay}
17: $Path_{r2r} \leftarrow WRR(s^*_{r,src}, s^*_{r,dst})$ {Relay to relay}
18: $Path_{r2d} \leftarrow \text{trivial}(s^*_{r,dst}, s_{dst})$ {Relay to destination}
19: $Path \leftarrow [Path_{s2r}, Path_{r2r}, Path_{r2d}]$

C. PolarDisCo Evaluation

We evaluate the delay of *PolarDisCo* and *DisCoRoute* for 5,000 pairs of satellites in the polar area when we set a zero processing delay on the 96.9°: 2000/40/21 constellation. To perform a fair comparison, we select random pairs that satisfy the conditions from Equation 2 and Equation 4 so that *PolarDisCo* does not revert to *DisCoRoute*. Our evaluation shows that we can achieve down to 7.9 ms, corresponding to 14.8% gain in delay. In the following, we analyze the performance of our approach when we relax the constraints on the selection of pairs.



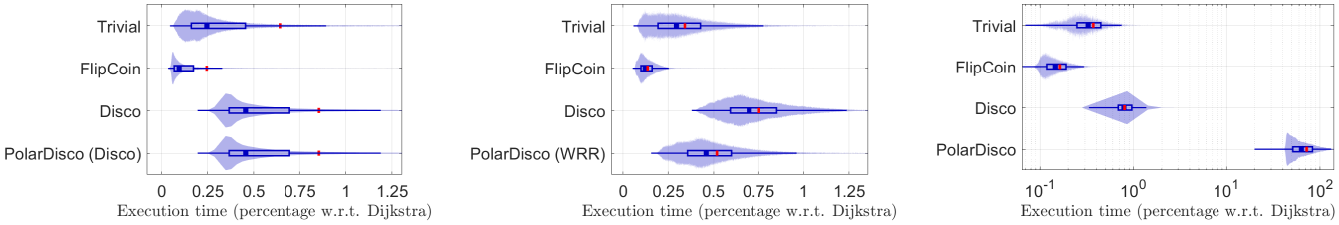
(a) Probability of applying *PolarDisCo*

(b) *PolarDisCo* vs. *DisCoRoute*

Figure 6: *PolarDisCo* and *DisCoRoute* delay comparison when applied to source and destination pairs of satellites in the polar region: (a) the probability of applying *PolarDisCo* depends on the latitude threshold of the considered polar region and amounts to up to 33.0% of the number of tested source-destination pairs; (b) *PolarDisCo* outperforms *DisCoRoute* in the polar region in most of the cases.

a) Frequency of Application of PolarDisCo: In Figure 6, we evaluate the probability of applying *PolarDisCo* in the polar region when we increase the latitude ϕ of the considered satellites while ignoring the hops condition Equation 4 in the pair selection. Although in most cases, *PolarDisCo* reverts to *DisCoRoute*, as explained in Section V-B, the benefits of *PolarDisCo* are evident in up to 33.0% of the polar cases.

b) Robustness to Hyper-parameters Choice: We evaluate the robustness of the proposed *PolarDisCo* algorithm concerning the following hyper-parameters: (*i*) latitude threshold of application of *PolarDisCo*, (*ii*) latitude threshold of application of WRR routing, (*iii*) latitude threshold for relay satellites in the polar region. Our analysis in Figure 6b shows that



(a) Outside Polar Zones (b) WRR zone (c) *PolarDisCo* zone

Figure 7: Computational time in different regions

PolarDisCo achieves consistent performance while varying such thresholds in an interval of 5 degrees around the chosen metric. In particular, *PolarDisCo* provides more than 2 ms decrease in average delay compared to *DisCoRoute* in such settings.

c) Computational Time: Here, we evaluate the computational time of the different proposed algorithms on the machine described in Section II-C. The distribution of the algorithms' execution time has been estimated by selecting 150,000 random source-destination satellite pairs outside the polar area, in the *PolarDisCo* area, and in the WRR zone, as depicted in Figure 7. Despite *DisCoRoute* being slightly more complexity demanding, all the *MinHopCount*-based algorithms exhibit much lower complexity than *Dijkstra-Distance*, with their worst-case execution times reaching only about 1.2% of *Dijkstra-Distance*'s. *PolarDisCo* provides lower computational times in the WRR zone, since it employs WRR to connect source and destination. However, it needs higher computational resources in the *PolarDisCo* zone since it requires computing a metric for each pair of k relay nodes.

VI. Conclusion

In this work, we analyzed routing algorithms tailored for LEO and VLEO satellite constellations. Our sensitivity analysis across multiple parameters demonstrated the critical impact of constellation geometry on routing efficiency, highlighting unique opportunities for low-latency paths, particularly for constellations in VLEO. Furthermore, we have demonstrated that while the routing solution does not directly depend on the constellation's altitude, the average end-to-end delay increases linearly with it. Also, the increased fleet size required for constellations in VLEO plays a crucial role, arguing for processing-efficient decentralized routing routines. Specifically for the critical polar regions, we introduced the *PolarDisCo* algorithm, designed to overcome the limitations of existing solutions like *DisCoRoute*. Experimental results revealed that *PolarDisCo* reduces end-to-end delay by up to 14.8%, underscoring its potential for enhancing communication efficiency in these challenging environments. These findings pave the way for more effective routing strategies in next-generation satellite networks spanning LEO and VLEO shells, particularly for applications requiring reliable polar coverage.

References

[1] W. Chen, X. Lin, J. Lee, *et al.*, "5G-advanced toward 6G: Past, present, and future," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 6, pp. 1592–1619, 2023. DOI: 10.1109/JSAC.2023.3274037.

[2] *6G: The next horizon*, White Paper, Huawei, Jan. 2022. [Online]. Available: https://www.huawei.com/en/huaweitech/future-technologies/6g-white-paper.

[3] N. Crisp, P. Roberts, F. Romano, *et al.*, "System modelling of very low earth orbit satellites for earth observation," *Acta Astronautica*, vol. 187, pp. 475–491, 2021. DOI: 10.1016/j.actaastro.2021.07.004.

[4] H. Luo, X. Shi, Y. Chen, *et al.*, "Very-low-earth-orbit satellite networks for 6G," 2022. [Online]. Available: https://www.huawei.com/en/huaweitech/future-technologies/very-low-earth-orbit-satellite-networks-6g.

[5] N. H. Crisp, P. C. E. Roberts, S. Livadiotti, *et al.*, "The benefits of very low earth orbit for earth observation missions," *Progress in Aerospace Sciences*, vol. 117, p. 100 619, 2020. DOI: 10.1016/j.paerosci.2020.100619.

[6] V. Ray, T. E. Berger, Z. C. Waldron, *et al.*, "The impact of space weather on very low earth orbit (VLEO) satellites," in *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, 2022.

[7] Q. Xiaogang, M. Jiulong, W. Dan, L. Lifang, and H. Shaolin, "A survey of routing techniques for satellite networks," *Journal of Communications and Information Networks*, vol. 1, no. 4, pp. 66–85, 2016. DOI: 10.11959/j.issn.2096-1081.2016.058.

[8] G. Chen, S. Wu, Y. Deng, J. Jiao, and Q. Zhang, "VLEO satellite constellation design for regional aviation and marine coverage," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 1, pp. 1188–1201, 2024. DOI: 10.1109/TNSE.2023.3321600.

[9] Z. Niu, H. Yang, Q. Yao, *et al.*, "Reliable low-latency routing for VLEO satellite optical network: A multi-agent reinforcement learning approach," *IEEE Internet of Things Journal*, 2024. DOI: 10.1109/JIOT.2024.3457498.

[10] G. Stock, J. A. Fraire, and H. Hermanns, "Distributed on-demand routing for LEO mega-constellations: A starlink case study," in *ASMS/SPSC 2022*, 2022, pp. 1–8. DOI: 10.1109/ASMS/SPSC55670.2022.9914716.

[11] J. G. Walker, "Satellite constellations," *Journal of the British Interplanetary Society*, vol. 37, no. 12, pp. 559–572, 1984.

[12] A. H. Ballard, "Rosette constellations of earth satellites," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-16, no. 5, pp. 656–673, 1980. DOI: 10.1109/TAES.1980.308932.

[13] Q. Chen, G. Giambene, L. Yang, C. Fan, and X. Chen, "Analysis of inter-satellite link paths for LEO mega-constellation networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2743–2755, 2021. DOI: 10.1109/TVT.2021.3058126.

[14] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. DOI: 10.1007/BF01386390.

[15] Space Exploration Holdings, LLC, *SpaceX non-geostationary satellite system: Attachment A, FCC IBFS SAT-MOD-20190830-00087*, Aug. 2019. [Online]. Available: https://fcc.report/IBFS/SAT-MOD-20190830-00087/1877671.

[16] Space Exploration Holdings, LLC, *SpaceX non-geostationary satellite system: Attachment A, FCC IBFS SAT-AMD-20210818-00105*, Aug. 2021. [Online]. Available: https://fcc.report/IBFS/SAT-AMD-20210818-00105/12943362.