Distributed On-Demand Routing for LEO Mega-Constellations: A Starlink Case Study

Gregory Stock, Juan Andrés A Fraire, Holger Hermanns

Distributed On-Demand Routing for LEO Mega-Constellations: A Starlink Case Study

Gregory Stock Juan A. Fraire Holger Hermanns
Saarland University – Computer Science, Saarland Informatics Campus, 66123 Saarbrücken, Germany
{stock, juanfraire, hermanns}@depend.uni-saarland.de

***Abstract*—The design and launch of large-scale satellite networks create an imminent demand for efficient and delay-minimising routing methods. With the rising number of satellites in such constellations, pre-computing all shortest routes between all satellites and for all times becomes more and more infeasible due to space and time limitations. Even though distributed on-demand routing methods were developed for specific LEO satellite network configurations, they are not suited for increasingly popular mega-constellations based on Walker Delta formations.**

The contributions of this paper are twofold. First, we introduce a formal model that mathematically captures the time-evolving locations of satellites in a Walker Delta constellation and use it to establish a formula to compute the minimum number of ISL hops between two given satellites. In the second part, we present an on-demand hop-count-based routing algorithm that approximates the optimal path while achieving superior performance compared to classical shortest-path algorithms like Dijkstra.

I. Introduction

Recent technology advances have led to a rapidly increasing number of satellite launches into low Earth orbit (LEO). Most of these satellites are part of huge mega-constellations that aim at providing global and fast communication, e. g. for internet access. Since most constellations scheduled for future deployment will rely on inter-satellite links (ISLs), packets will usually need to take a high number of hops when travelling from source to destination. The huge number of satellites adds to the complexity of routing in such a network. There is thus a need for specialised and efficient routing algorithms, especially since space networks have to compete with existing and already well-understood ground networks where the topology is more stable compared to their counterparts in space. Besides offering global coverage, satellite operators aim at providing higher speeds and lower latencies compared to traditional broadband. For this, it is critical that one can quickly determine routes that minimise the delay. Part of the solution lies already in the design phase of the constellation where a lot of room for optimisation exists. Routing, however, remains an important problem once the constellation is in orbit and operable. Packets need to travel large distances in space and pass several satellites that relay the traffic from one satellite to its neighbour. It is, therefore, generally beneficial to select short routes. However, since every extra ISL relay adds additional processing costs, it is even more important to keep the number of hops as low as possible. This raises the important challenges of how to calculate the number of hops needed to connect two satellites in the constellation and how to use these results when it comes to routing.

This paper first provides a mathematical model to express and work with satellite positions in a Walker Delta constellation, a popular constellation type consisting of two locally separate overlapping meshes, an ascending and a descending one (Section II). Then, this model is used in Section III to derive a formula for the minimum number of ISL hops required to connect two satellites. These results are the basis for two new on-demand routing algorithms that produce routes with the minimum possible number of hops (Section IV). The algorithms have almost no overhead and work in a distributed way, as neither an exhaustive exploration nor a centralised pre-computation of the path is needed. The first algorithm simply selects hops probabilistically while assuring hop-minimality. The second algorithm uses a heuristic to generate near-optimal approximations of the best route but still comes at almost no cost. This heuristic is based on observations on the distances spanned by inter- and intra-plane hops that we introduce in Section V. It is optimal, i. e. it is guaranteed to find the overall shortest route, for the base case of constellation parameters. Finally, we report (Section VI) on a thorough empirical evaluation spanning simulations of artificial as well as real-world constellations, with a focus on Starlink. The proposed algorithms are significantly faster than the state of the art while maintaining competitiveness regarding the quality of the routes. We come to the conclusion that the hop-count-based algorithms are superior to classical routing algorithms based solely on shortest paths.

II. Background

A. Existing Routing Algorithms

Routing, in general, is a well-studied topic that is also addressed by many research papers in the context of space surveyed in [1], [2]. Yet, most algorithms have been developed quite some time ago and have had the needs of those times as motivations. For example, they only support polar constellations (e. g. Walker Star) or are designed for small ATM-like packets which have minimal relay time so that the hop count is not so important [3]. This means that these algorithms are not necessarily suitable or optimal for upcoming mega-constellations. This paper focuses on a type of constellation called *Walker Delta* that has recently found popularity in the design of upcoming mega-constellations according to regulatory filings. The satellites in such a constellation follow circular orbits and are partitioned into different orbital planes. Before

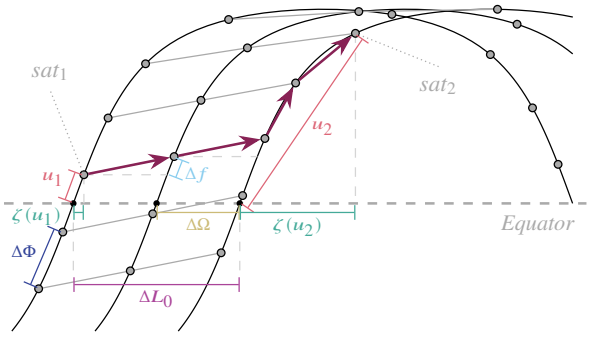

Fig. 1: Ground plot of a section of a constellation annotated by the various modelling parameters. All indicated parameters are given as angles measured from the centre of Earth.

we mathematically describe these constellations in more detail, we first consider the modelling of individual satellites.

B. Satellite Model

The current position of a satellite is typically represented using the six Keplerian elements, also known as classical orbital elements. Most important are the *longitude of the ascending node* Ω and the *true anomaly* ν. In the following, we will usually refer to the *initial longitude of the ascending node* $L_0 \in [-\pi, \pi[$ at some Epoch, which is a constant and independent of the current time t. The current longitude of the ascending node Ω at time t is given by $\Omega = L_0 - \omega_E \cdot t$, where ω_E is the angular speed of Earth's rotation. Since the orbits are circular (i. e. eccentricity $e = 0$), the true anomaly is undefined, as the periapsis cannot be uniquely determined. Therefore, the *argument of latitude* (or phase angle) $u \in [-\pi, \pi[$ is used instead, which is the angle between the ascending node and the satellite and basically defines the position of the satellite in its orbit. We say that a satellite is *ascending*, i. e. flying in north-east direction, if $u \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and call it otherwise *descending*, i. e. flying towards the south-east. The remaining orbital elements, i. e. *semi-major axis* a and *inclination* α, are equal for all satellites and therefore considered global constants of the constellation. Note that the semi-major axis of the orbit is equal to its radius due to its circular shape. Further, note that we define a satellite's *altitude* h relative to the surface of the Earth, i. e. the orbit radius is equal to the sum of h and Earth's semi-major axis $r_a = 6378.137\,\text{km}$ as specified in the WGS84 reference system [4].

While it is very descriptive and convenient to model the position of a single satellite using these basic orbital parameters, it is usually not very useful when comparing the positions of different satellites, e. g. calculating their distance. Therefore, we show in the following how the argument of latitude u and the (initial) longitude of the ascending node L_0 can be converted first to geodetic coordinates (latitude & longitude) and then to a cartesian coordinate system known as ECEF (Earth-centered, Earth-fixed).

1) Keplerian to Geodetic: Geodetic coordinates are specified using *latitude* φ, *longitude* λ, and the *height* h. The conversion of a satellite position to geodetic coordinates is given by:

$$\varphi = \arcsin(\sin\alpha \cdot \sin u) \in [-\alpha, \alpha]$$
$$\lambda = \mathcal{N}\big(L_0 - \omega_E \cdot t + \zeta(u)\big) = \mathcal{N}\big(\Omega + \zeta(u)\big) \in [-\pi, \pi[$$

Here, $\zeta(u)$ indicates the *longitude difference* of a satellite to its ascending node (see Fig. 1):

$$\zeta(u) = \arctan(\cos\alpha \cdot \tan u) + \begin{cases} 0 & \text{asc. segment} \\ \pi & \text{desc. segment} \end{cases}$$

$\mathcal{N}(x) = \big((x + \pi) \bmod 2\pi\big) - \pi$ is a normalisation function that ensures that the resulting values are within the desired interval $[-\pi, \pi[$. An important application of geodetic coordinates is the calculation of the sub-satellite point. It is defined as the point where a straight line from the centre of the Earth to the satellite intersects with the surface of the Earth. Since the geodetic coordinate system is a spherical system, the sub-satellite point of a satellite has the same latitude and longitude as the satellite, i. e. only its altitude differs.

2) Geodetic to Cartesian: The *Earth-centered, Earth-fixed* coordinate system (ECEF) is a geocentric system that uses Cartesian coordinates. This representation is well suited to compute distances between two objects in space. Converting geodetic coordinates to Cartesian coordinates (X, Y, Z) can be done as follows:

$$\big(\underbrace{(r_a + h)\cos\varphi\cos\lambda}_{X},\ \underbrace{(r_a + h)\cos\varphi\sin\lambda}_{Y},\ \underbrace{(r_a + h)\sin\varphi}_{Z}\big)$$

C. Walker Delta Constellation

A Walker Delta constellation consists of P orbital planes that are evenly spaced around the Equator. Each of these planes contains Q evenly spaced satellites. All satellites follow a circular orbit with the same inclination α and altitude h. A Walker Delta constellation is often formally described by $\alpha : PQ/P/F$ where F indicates the relative spacing between satellites in adjacent planes (see below). Throughout this paper, we use (o, i) to denote the i-th satellite in orbital plane o.

The *RAAN difference* (i. e. difference in right ascension of the ascending node) between adjacent planes $\Delta\Omega = \frac{2\pi}{P} \in [0, 2\pi]$ specifies how far apart neighbouring planes are from each other and only depends on the number of orbital planes. Within an orbital plane, the *phase difference*, i. e. the difference in argument of latitude, between adjacent satellites $\Delta\Phi = \frac{2\pi}{Q} \in [0, 2\pi]$ can be computed from the number of satellites per plane. Finally, the *phase offset* $\Delta f = \frac{2\pi F}{PQ} \in [0, 2\pi[$ between satellites in adjacent planes specifies the difference in argument of latitude between two horizontal neighbours. This value is usually given in terms of a *phasing factor* $F \in \{0, \dots, P-1\}$. In contrast to arbitrary phase offsets, this factor ensures that the sum of phase offsets for all P planes, i. e. $P \cdot \Delta f$, is always a multiple of $\Delta\Phi$. This property is required for the constellation to be symmetric, i. e. to have the same Δf for every satellite. See Fig. 1 for a visualisation of these parameters. Satellite (o, i) can now be described by $\big(L_0 = \mathcal{N}(o \cdot \Delta\Omega),\ u = \mathcal{N}(o \cdot \Delta f + i \cdot \Delta\Phi)\big)$ for $0 \le o < P$ and $0 \le i < Q$.

a) Inter-Satellite Links: Throughout this paper, we assume that each satellite can establish four ISL links to its immediate neighbours, i. e. an intra-plane link to the successor and predecessor in its own orbital plane and an inter-plane link each to the neighbour in the left and right plane. Walker Delta constellations are typically deployed in low-inclination orbits (e. g. 53.2° for Starlink's inner shell) where the Doppler impairments in cross-plane links can be coped with at the highest latitudes. Formally, the predecessor and successor of satellite (o, i) are given by $(o,\ (i-1) \bmod Q)$ and $(o,\ (i+1) \bmod Q)$, respectively. The left neighbour is $(o-1,\ i)$ if $o \neq 0$ and $(P-1,\ (i-F) \bmod Q)$ otherwise. Analogously, the right neighbour is $(o+1,\ i)$ if $o \neq P-1$ and $(0,\ (i+F) \bmod Q)$ otherwise.

III. Minimum Hop Count

In this section, we want to show how the geodetic positions of the satellites in the constellation can be leveraged to compute the minimum number of ISL-hops required to connect two satellites. This section is based on previous work by Chen et al. in which a theoretical model to estimate the ISL hop count was presented [5]. We use these results as a foundation and augment them by our findings. We do not consider how to select adequate access satellites for ground users but instead work with the assumption that the two satellites for which the hop count or route is to be calculated are given.

A. Hop Count Model

In the following formulas, we often use $\lfloor \cdot \rceil$ to indicate that a number is rounded to the nearest integer according to the rounding function known as *commercial rounding* that rounds half away from zero, i. e. $\lfloor x \rceil = \mathrm{sgn}(x)\ \lfloor |x| + 0.5 \rfloor$. Note, however, that in our theoretical context, rounding is not strictly necessary as for all quotients the divisor is always a factor of the respective dividend.

Computing the minimum number of inter-plane hops H_h is fairly straightforward. It is given by the horizontal distance that must be travelled to get from the initial orbital plane to the destination plane. Therefore, it only depends on the difference between the longitudes of the respective ascending nodes:

$$\Delta L_0 = (L_{0,2} - L_{0,1}) \bmod 2\pi \in [0, 2\pi[$$

ΔL_0 is the longitudal angle that must be covered when going from the orbital plane of the source satellite to the plane of the destination in east direction. Therefore, the RAAN difference in west direction is $2\pi - L_0$. Since each hop from one plane to the next covers an angle of $\Delta\Omega$, the total number of inter-plane hops in east or west direction is given by:

$$H_h^{\leftarrow} = \left\lfloor \frac{2\pi - \Delta L_0}{\Delta\Omega} \right\rceil \qquad H_h^{\rightarrow} = \left\lfloor \frac{\Delta L_0}{\Delta\Omega} \right\rceil$$

Next, we need to look at the phase angle differences between the two satellites. An intra-plane hop (to the successor satellite) adds $\Delta\Phi$ to the phase angle, while an inter-plane hop (towards the east) increases the phase angle by Δf. Formally, when taking only hops to the successor and right neighbour, the following relationship holds:

$$u_2 = u_1 + (H_h^{\rightarrow} \cdot \Delta f) + \underbrace{(H_v^{\nearrow} \cdot \Delta\Phi)}_{\Delta\vec{u}}$$

Since we want to compute the number of intra-plane hops H_v, we have to compute the fraction $\Delta\vec{u}$ of the phase angle difference that will be covered by intra-plane hops. We again need to distinguish between two directions:

$$\Delta\vec{u} = (u_2 - u_1 - H_h^{\rightarrow} \cdot \Delta f) \bmod 2\pi$$
$$\Delta\overleftarrow{u} = (u_2 - u_1 + H_h^{\leftarrow} \cdot \Delta f) \bmod 2\pi$$

Finally, we can do similar calculations as for H_h to compute the directional intra-plane hop counts:

$$H_v^{\nwarrow} = \left\lfloor \frac{\Delta\overleftarrow{u}}{\Delta\Phi} \right\rceil \qquad H_v^{\nearrow} = \left\lfloor \frac{\Delta\vec{u}}{\Delta\Phi} \right\rceil$$
$$H_v^{\swarrow} = \left\lfloor \frac{2\pi - \Delta\overleftarrow{u}}{\Delta\Phi} \right\rceil \qquad H_v^{\searrow} = \left\lfloor \frac{2\pi - \Delta\vec{u}}{\Delta\Phi} \right\rceil$$

The minimum hop count is then just given by the minimum of the possible combinations:

$$\min\{H_h^{\leftarrow} + H_v^{\nwarrow},\ H_h^{\leftarrow} + H_v^{\swarrow},\ H_h^{\rightarrow} + H_v^{\nearrow},\ H_h^{\rightarrow} + H_v^{\searrow}\}$$

Note that this formula also contains indicators for the two directions in which to travel to achieve the minimum number of hops.

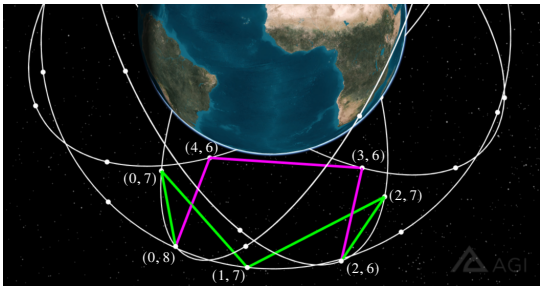

Fig. 2: A Walker Delta constellation 60° : 50/5/2 pinpointing the difference where the original hop count formula produces suboptimal results.

B. Enhancement over Existing Formula

As mentioned, the above derivations are inspired by results established by Chen et al. [5]. The latter are based on the assumption that *"if the path on a given direction is too long (e. g. $H_h > {}^{P}/_{2}$), the packets will go through the opposite direction."* The above results relinquish this assumption, because it may induce unnecessarily high hop counts. As a concrete example, Fig. 2 shows a Walker Delta constellation (60° : 50/5/2) that visualises this fine point. The constellation contains $P = 5$ planes which means that according to the assumption, for all routes with more than two inter-plane hops, there should be an alternative route with $H_h \leq {}^{P}/_{2}$ and $H_v \leq {}^{Q}/_{2}$ that contains

fewer or equally many hops. Now consider the two routes from $(0, 8)$ to $(2, 6)$, for which the original work of Chen et al. [5], gives a minimum hop count of four. The corresponding route is depicted in green. It has one intra-plane hop, then two inter-plane hops and one more intra-plane hop. The purple route has a hop count of only three (which is the value returned by our formula). It consists of three inter-plane hops. Notably, it is also shorter in total distance spanned.

C. Hop Count Evaluation

We validated the correctness of our formula by empirically comparing it to an algorithm that derives the hop count from the satellites' identifiers (o, i). Then, we evaluated the formula and analysed the differences on the Starlink constellation (see Section VI-A). Across all possible combinations of satellite pairs, the hop count using Chen's formula was unnecessarily high in about 2.7 % of the pairs (1.26 % one additional hop, 1.01 % three additional hops, 0.44 % five additional hops).

Furthermore, we compared the hop counts with those of the shortest-distance routes which in turn were computed using Dijkstra's algorithm. Interestingly, the latter routes sometimes contain one more hop than the minimum (≈ 1 % of the pairs). However, whenever this is the case, both hop count formulas agree on the count.

IV. Routing Algorithms

The second part of this paper considers different solutions for solving the routing problem. First, we introduce Dijkstra's algorithm as the classical algorithm for solving shortest path problems. Next, we argue why considering a metric based on hop count can be superior to just considering the length, i. e. the total distance spanned by a route. Finally, we suggest and evaluate new routing algorithms that integrate the minimum hop count.

A. Dijkstra's Shortest Path

Dijkstra's algorithm [6] is a well-known algorithm and the de-facto standard for solving shortest path problems. It is also commonly used in the domain of network routing and exists in various flavours. In this paper, we are only interested in solving the single-pair shortest path problem. We consider the Euclidean distance between satellites as a metric and use a min-heap to store the unvisited nodes. The pseudocode is shown in Algorithm 1. Note that to compute the shortest paths from the source to all other nodes, it suffices to remove the lines 12 to 14 which serve as a shortcut to terminate the algorithm once the destination node has been reached.

Once the algorithm finishes, the total distance of the shortest path from source to destination is $dist[dst]$. However, we are usually also interested in the route itself rather than just its length. Therefore, the code displayed in Algorithm 1 also remembers for each visited node its predecessor node on the shortest path to it. The reconstruction algorithm starts at the destination node and basically follows the pointers to the respective previous nodes until it arrives at the source node, keeping track of all nodes that were passed.

Algorithm 1 Dijkstra's Shortest Path

```
1  procedure DIJKSTRA(Const., src, dst)
2      for all satellites v ∈ Constellation do
3          dist[v] ← ∞                          ▷ unknown distance to v
4          prev[v] ← ⊥                          ▷ predecessor of v
5          visited[v] ← false
6      end for
7      dist[src] ← 0
8      Q ← HEAPIFY({(v, dist[v]) | v ∈ Const.})
9      while Q is not empty do
10         (u, d) ← Q.POP()                     ▷ pop sat. u with min. distance d
11         visited[u] ← true
12         if u = dst then
13             break                            ▷ shortest path found
14         end if
15         for all neighbors v of u do
16             if ¬visited[v] then
17                 alt ← d + EUCLIDEAN(u, v)
18                 if alt < dist[v] then
19                     dist[v] ← alt            ▷ found shorter alternative
20                     prev[v] ← u
21                     Q.DECREASEKEY(v, alt)    ▷ update priority
22                 end if                       of satellite v
23             end if
24         end for
25     end while
26     return dist[], prev[]
27 end procedure
```

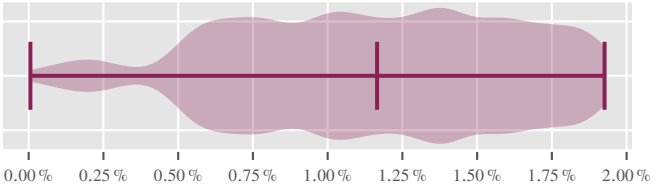

Fig. 3: Violin plot showing how much longer the shortest route with minimum number of hops is compared to the overall shortest route for Starlink.

We use Dijkstra's algorithm in the following as a baseline for finding shortest routes between pairs of satellites and use it to compare the performance of other algorithms.

B. Shortest Length vs. Minimum Hops

In this section, we describe how the routing method can be optimised by exploiting the minimum hop count formula. This improvement is based on the observation that the shortest route usually also has the lowest possible number of hops. Section III-C already showed that there are some exceptions to this where, in some rare cases, the shortest route can contain more hops than the minimum. Fig. 3 is a violin plot for the Starlink constellation which shows how much longer the shortest route with minimum number of hops is compared to the overall shortest path. It considers only the $\approx$1 % of satellite pairs where the shortest path has more hops than the minimum. The largest difference is a $\approx$2 % longer route (which amounts to around 1284 km in absolute terms). In all cases, there is only one additional hop.

However, we argue that the number of hops is actually the metric that should be minimised first. Mega-constellations will need to carry large amounts of data, which is only possible with longer data packets, e. g. jumbo Ethernet frames. As the signals travel with the speed of light, it is clear that slightly shorter routes do not compensate for the overhead of an additional hop. For example, the transmission time of a packet of size 65 535 bytes (largest possible IPv4 payload) at an ISL rate of 1 Gbps is about 500 µs. Even if faster ISL data rates are leveraged, there is internal packetisation and queuing delay which can easily add up to several milliseconds of on-board processing, even when using highly efficient ATM fabrics [7]. A single hop is thus comparable with the propagation delay of 1000 km distance at the speed of light (299 792 km/s) totalling 3.3 ms.

Knowing the minimum number of hops in both dimensions and their directions between two satellites allows the routing algorithm to restrict the exploration: The search space can be reduced significantly, forming a spherical rectangle where the source and destination satellites are at opposing corners. Note that in this rectangle, every possible route from the source to the destination has the same number of hops (similar to a Manhattan Street Network [8]).

The fact that the search space is now a two-dimensional grid has the side-effect that it becomes a directed acyclic graph (DAG). For DAGs, there is a more efficient algorithm than Dijkstra. DAGshort is based on topological sorting and runs in $O(V+E)$ time [9]. A topological order is a linear ordering of the graph's nodes where for every directed edge in the graph, the source is sorted before the destination in the ordering. There exists a trivial ordering for the two-dimensional grid of ISL links: Starting at the source node, all vertices in the grid can be enumerated line by line, i. e. sorting them first by their vertical distance to the source and then by their horizontal distance.

For the sake of completeness, we want to mention yet another possible approach, namely a modification of Dijkstra, denoted DijkstraHops, that computes the shortest route amongst the ones with the minimum number of hops. For this, it suffices to store tuples $(hops, distance)$ on the heap and compare them using a lexicographic order.

C. Probabilistic Routing

We now want to quantify the difference between the best and worst possible route in such a spherical rectangle. The question is whether it actually pays off to invest in compute-intensive routing algorithms or whether it suffices to (randomly) send the packet via any route with minimum number of hops. For this, we analyse an algorithm CoinFlipRoute that, given a source and destination, first computes the minimum hop count and directions using the formula from Section III-A. Then, it flips a coin at each satellite to randomly decide in which of the two directions the packet should continue. The only difficulty is to keep track of the vertical and horizontal hops to restrict the route to the spherical rectangle. When a packet has, for example, travelled all of its horizontal hops, the route is forced to be completed with the remaining number of vertical hops.

Algorithm 2 DisCoRoute (Cases A2A & D2D)

```
1  procedure DisCoRouteA2A(Const., src, dst)
2      H_h, H_v ← MinHopCount(Const., src, dst)
3          ▷ w. l. o. g. s_{0,0} = src, s_{H_h,H_v} = dst, dst is north east of src
4      route_s ← [src]
5      route_t ← [dst]
6      i ← 0
7      j ← H_h
8      for H_h many times do
9          reward_s ← |φ_{i,0} + φ_{i+1,0}|
10         reward_t ← |φ_{j,H_v} + φ_{j-1,H_v}|
11         if reward_s < reward_t then
12             route_t ← s_{j-1,H_v} :: route_t
13             j ← j − 1
14         else
15             route_s ← route_s :: s_{i+1,0}
16             i ← i + 1
17         end if
18     end for
19     assert i = j          ▷ s_{i,0} and t_{j,H_v} are on same orbital plane
20     if H_v = 0 then       ▷ is route_t[0] also last element of route_s?
21         route_t ← route_t[1:]                    ▷ remove first element
22     else
23         route_s ← route_s + [s_{i,1}, ..., s_{i,H_v-1}]
24     end if
25     return route_s + route_t
26 end procedure
```

Notably, the approach of locally flipping a coin is not the same as enumerating all possible routes in the rectangle and selecting one of them uniformly at random. This is because some satellites are restricted in their choice, and therefore the possible routes do not all have the same probability.

As we will see in the evaluation (Section VI), the CoinFlipRoute approach turns out to perform quite well in practice and is very cheap to compute. However, we present an algorithm in the next section that is still easy to compute but aims at calculating more reasonable routes.

D. DisCoRoute

DisCoRoute is a **dis**tributed routing algorithm for mega-**co**nstellations that exploits the insights found so far. The key strength of this algorithm is that it is very cheap to compute compared to Dijkstra. The algorithm produces near-optimal solutions that outperform the probabilistic approach. In the following, we first introduce the idea and requirements of the algorithm, then describe how the algorithm works, and in Section VI, we show that the loss of optimality is not significant in practice. The algorithm is rooted in two insights:

1) The length of an *intra*-plane hop is always constant in the constellation, independent of the satellite's positions.
2) The length of an *inter*-plane hop should decrease the further it is away from the Equator.

Therefore, the main idea is to cleverly distribute the inter-plane hops so that they happen as close as possible to the poles.

Similar to the probabilistic algorithm, we first need to compute the minimum hop count and the respective directions from source to destination. Then, the algorithm distinguishes two cases depending on the flying direction of the two satellites.

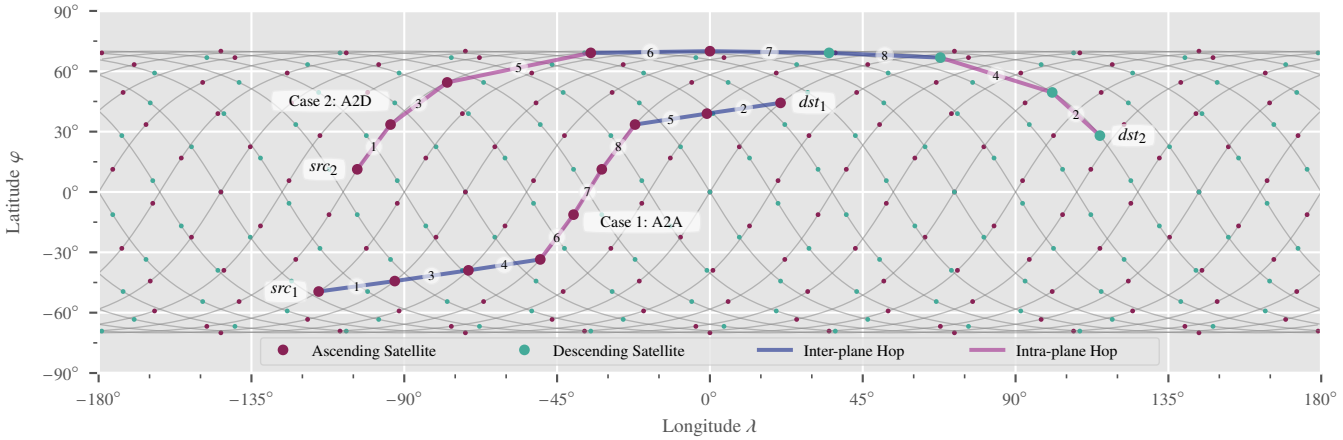

Fig. 4: Ground track of a Walker 70° : 300/20/5 constellation showing an example route for each of the two cases of DisCoRoute.

Algorithm 3 DisCoRoute (Cases A2D & D2A)

```
1  procedure DisCoRouteA2D(Const., src, dst)
2      H_h, H_v ← MinHopCount(Const., src, dst)
3      ▷ w.l.o.g. s_{0,0} = src, s_{H_h,H_v} = dst, dst is north east of src
4      route_s ← [src]
5      route_t ← [dst]
6      i ← 0
7      j ← H_v
8      for H_v many times do
9          reward_s ← |φ_{0,i} + φ_{0,i+1}|
10         reward_t ← |φ_{H_h,j} + φ_{H_h,j−1}|
11         if reward_s < reward_t then
12             route_s ← route_s :: s_{0,i+1}
13             i ← i + 1
14         else
15             route_t ← s_{H_h,j−1} :: route_t
16             j ← j − 1
17         end if
18     end for
19     assert i = j    ▷ s_{0,i} and t_{H_v,j} are reachable via horiz. hops
20     if H_h = 0 then    ▷ is route_t[0] also last element of route_s?
21         route_t ← route_t[1:]    ▷ remove first element
22     else
23         route_s ← route_s + [s_{1,i}, ..., s_{H_h−1,i}]
24     end if
25     return route_s + route_t
26 end procedure
```

a) Case 1: Asc-to-Asc / Desc-to-Desc: Assume w.l.o.g. that both satellites are ascending. The idea is to distribute all inter-plane hops between the beginning and end of the route, having all intra-plane hops in the middle. We chose the partition that maximises the overall distance of all inter-plane hops from the Equator. The pseudocode is shown in Algorithm 2. It starts by constructing the route simultaneously from the source and destination. Then, it looks at the absolute value of the sum of latitudes of the inter-plane hop that starts at the source and of the one that ends at the destination. The one with a larger value is added to the (respective) route. This selection method is repeated for the number of horizontal hops many times. Afterwards, the two route segments only need to be connected by the given number of intra-plane hops (potentially zero).

b) Case 2: Asc-to-Desc / Desc-to-Asc: This case is roughly the inverse of the first case. Routes from ascending to descending satellites must always go close to a pole since this is the only location where there is a link between an ascending and a descending satellite. This means that the general rule should be to partition the intra-plane hops to the beginning and end of the route and have all inter-plane hops in the middle, as close to the pole as possible. The pseudocode in Algorithm 3 has a similar structure as the first case. The only significant difference is the selection criterion in line 11 that chooses an intra-plane hop on the side where the absolute value of the latitude sum of the involved satellites is smaller.

Fig. 4 shows an example for both cases. The lower route corresponds to the first case. It starts with an ascending satellite in the southern hemisphere and ends with an ascending satellite in the northern one, meaning that it must cross the Equator. As one can see, the three inter-plane hops right at the beginning and the two at the end are distributed to maximise their overall distance from the Equator. Note that if both satellites were in the northern hemisphere, the route would not start with inter-plane hops but would perform all of them at the end. The route for the second case connects an ascending satellite with a descending one. We can see that the selected intra-plane hops allow the inter-plane hops to have maximal distance from the Equator, i. e. have the shortest length possible.

V. Hop Distances

This section covers the length calculations of inter- and intra-plane hops. Using the following theorems, it becomes trivial to prove that the approximative DisCoRoute algorithm can solve the shortest path problem exactly for Walker Delta constellations with zero phase offset $\Delta f = 0$.

Theorem 1 (Length of Vertical Hops): The travel distance of an intra-plane hop is always the same, no matter where the hop is performed in the constellation.

Theorem 2 (Length of Horizontal Hops): In a constellation with zero phase offset $\Delta f = 0$, the travel distance of an inter-plane hop is shortest closest to the poles and longest around the Equator.

Due to space limitations, the proofs are omitted in this paper, but they can be found in the technical report [10].

Our empirical evaluation suggests that Theorem 2 also applies to the more general case for non-zero phase offsets. Unfortunately, we were unable to formally prove this and to provide a closed formula for the length of an inter-plane hop. We nevertheless formulate our findings as two conjectures that describe when the inter-plane hop distance between two neighbouring satellites is maximal and minimal.

Conjecture 1 (Longest Horizontal Hop): An inter-plane hop between two neighbouring satellites has the longest distance when the phase angles of both satellites are equally distributed around the Equator. This means that the phase angles of the satellites are either $u_1 = -\Delta f/2$ and $u_2 = \Delta f/2$ or, analogously on the backside of the Earth, $u_1 = \pi - \Delta f/2$ and $u_2 = \pi + \Delta f/2$.

Conjecture 2 (Shortest Horizontal Hop): A horizontal hop is smallest when the phase angles of both satellites are equally distributed around their points with maximal latitude. This means that the phase angles of the satellites are either $u_1 = \pi/2 - \Delta f/2$ and $u_2 = \pi/2 + \Delta f/2$ or, analogously on the backside of the Earth, $u_1 = -\pi/2 - \Delta f/2$ and $u_2 = -\pi/2 + \Delta f/2$.

It is noteworthy to mention that these two cases have a convenient property when considering the absolute value of the two satellites' latitudes $|u_1 + u_2|$ (as we do in DisCoRoute): The longest distance of an inter-plane hop is achieved at the location where both satellites have the same latitude and their sum is maximal. In contrast, the distance is minimal when the latitudes have the same magnitude but with different signs, i. e. their sum is zero.

VI. Evaluation

In the last section of this paper, we provide the results of our empirical performance evaluation of the algorithms through simulations. We have implemented our model and algorithms both in Python and Rust. The measurements presented in this section are all produced using the Rust implementation as it is significantly faster than Python code. However, this only affects the run time measurements since, apart from that, the two implementations calculate identical results. All benchmarks were run on a Linux machine, equipped with an Intel Core™ i7-6700 CPU running at 3.40 GHz and 32 GB of main memory.

A. Setup: Starlink Constellation

Most of the following benchmarks are executed on the (first) orbital shell of the initial deployment phase of SpaceX's Starlink constellation. More precisely, the Starlink constellation that we consider is formally described as a Walker Delta 53.0°: 1584/72/39 at 550 km. These parameters are taken from publicly available information [11], except for the phasing factor $F = 39$ which is not explicitly mentioned and had to be estimated based on the available documents and by inspecting the publicly available data of satellites launched so far.

B. Run Time

The violin plot in Fig. 5 shows the run time distribution of the different algorithms. For each algorithm, we measure its execution time for all possible combinations of two satellites in the Starlink constellation. Note that the plot uses a logarithmic scale and that we use the average of ten runs for each pair. We observe that Dijkstra's algorithm is, as expected, the most expensive one. This is mainly due to the fact that Dijkstra explores neighbour satellites in all four directions while all other algorithms first compute the minimum hop count and then restrict the search space to the induced spherical rectangle. Comparing classical Dijkstra and DAGshort, we achieve a mean speedup of around 22×. Considering DisCoRoute, we even end up with a mean speedup of 158× compared to classical Dijkstra and still 7× compared to DAGshort. Interesting to note is that the CoinFlipRoute algorithm has a slightly lower mean than DisCoRoute but is slower in the worst case.

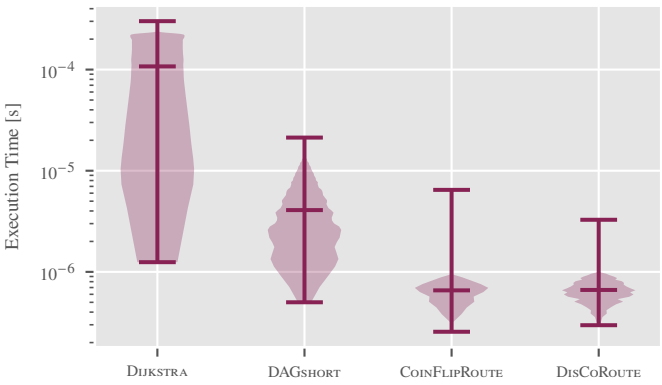

Fig. 5: Comparison of the execution times (in seconds) for each algorithm on all satellite pairs in the Starlink constellation.

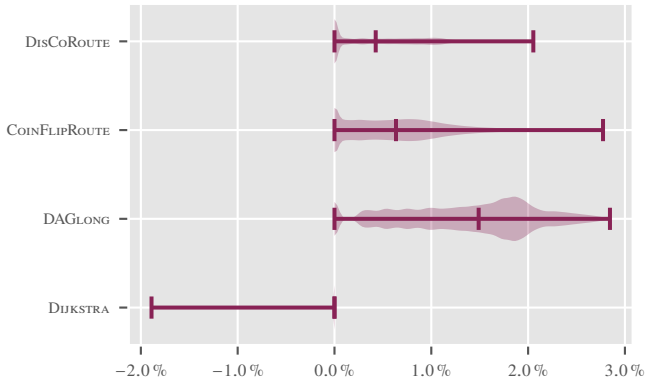

Fig. 6: Comparison of (relative) route lengths produced by each algorithm where the baseline is the shortest route with minimum hops, i. e. DAGshort.

C. Exactness

Unlike Dijkstra, the hop-count-based algorithms are all approximates and do not always find the overall optimal solution. This section is devoted to the analysis of the magnitude of suboptimality of these algorithms. For this, we look again at all pairs in the Starlink constellation and compute the lengths of all computed routes. We use the hop-count-based shortest path algorithm DAGshort as a baseline and compare the

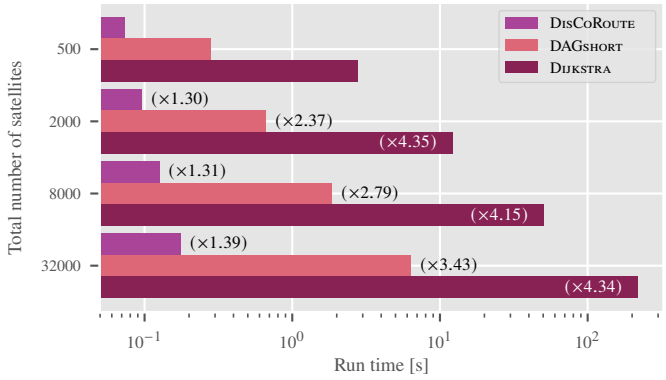

Fig. 7: Bar chart showing the scalability of the algorithms with increasing constellation size.

other algorithms relative to it. This has the effect that classical Dijkstra is the only algorithm that can be better than the baseline. For completeness, we also include an algorithm DAGlong that computes the length of the longest possible path inside the spherical rectangle induced by the hop counts. The results are depicted in the violin plot in Fig. 6.

As we have already seen in Section IV-B, Dijkstra can outperform the baseline in only a tiny fraction of cases. In the worst case, Dijkstra would be able to produce an up to 2 % shorter route. However, the mean of Dijkstra is just at around −0.02 %. Further, we see that DisCoRoute and CoinFlipRoute perform quite well in practice.

D. Scalability

As a final performance indicator, we evaluate how well the different algorithms scale when the number of satellites in the constellation increases. For this, we created four test-constellations 60° : 500/25/5, 60° : 2000/50/10, 60° : 8000/100/20, and 60° : 32 000/200/40. For each of them, we sampled 100 000 random satellite pairs and measured the execution time each algorithm takes. The results are depicted on a logarithmic scale in the bar chart in Fig. 7.

Note that the number of satellites quadruples in each step. The number in parentheses indicates the factor by which the run time increases compared to the previous case. It is clear from the figure that our DisCoRoute algorithm provides better scalability on huge mega-constellations compared to Dijkstra.

VII. Conclusion

This paper has started off with the question whether there exist simple routing schemes that optimise the number of hops as well as the travel distance between pairs of satellites in mega-constellations. Apart from a straightforward probabilistic forwarding scheme, we have introduced the DisCoRoute algorithm and have presented profound empirical studies discussing the pros and cons in comparison to the optimal solution, thereby shedding light on the relative impact of travel time and hop count minimisation. The results are overall very encouraging. We have focussed on Walker Delta constellations (like Starlink) but are exploring extensions to other types of constellations. Furthermore, we are currently embarking on extensions of this work, taking into account congestion and background traffic, as well as further evaluations (e. g. route length and latency).

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101008233 (MISSION).

References

[1] X. Qi, J. Ma, D. Wu, L. Liu, and S. Hu, "A survey of routing techniques for satellite networks," *J. Commun. Inf. Networks*, vol. 1, no. 4, pp. 66–85, 2016. doi: 10.1007/BF03391581.

[2] M. A. A. Madni, S. Iranmanesh, and R. Raad, "Dtn and non-dtn routing protocols for inter-cubesat communications: A comprehensive survey," *Electronics*, vol. 9, no. 3, 2020. doi: 10.3390/electronics9030482.

[3] H. Li and X. Gu, "Adaptive ATM routing in Walker delta satellite communication networks," in *1st Int. Symp. Syst. Control Aerosp. Astronautics*, 2006, pp. 368–373. doi: 10.1109/ISSCAA.2006.1627646.

[4] Defense Mapping Agency, "Department of defense world geodetic system 1984," DMA, TR 8350.2, 1991. [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA280358.pdf.

[5] Q. Chen, G. Giambene, L. Yang, C. Fan, and X. Chen, "Analysis of inter-satellite link paths for LEO mega-constellation networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 3, pp. 2743–2755, 2021. doi: 10.1109/TVT.2021.3058126.

[6] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959. doi: 10.1007/BF01386390.

[7] D. Cerovic, V. D. Piccolo, A. Amamou, K. Haddadou, and G. Pujolle, "Fast packet processing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3645–3676, 2018. doi: 10.1109/COMST.2018.2851072.

[8] N. F. Maxemchuk, "Routing in the manhattan street network," *IEEE Trans. Commun.*, vol. 35, no. 5, pp. 503–512, 1987. doi: 10.1109/TCOM.1987.1096802.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. MIT Press, 1989, pp. 536–538, Section 25.4, isbn: 0-262-03141-8.

[10] G. Stock, J. A. Fraire, and H. Hermanns, "Distributed on-demand routing for LEO mega-constellations: A starlink case study," *CoRR*, 2022. doi: 10.48550/arXiv.2208.02128. arXiv: 2208.02128.

[11] Space Exploration Holdings, LLC, *SpaceX non-geostationary satellite system: Attachment A*, FCC IBFS SAT-MOD-20190830-00087, Aug. 2019. [Online]. Available: https://licensing.fcc.gov/myibfs/download.do?attachment_key=1877671.