applied sciences

MDPI

Article

Dynamic Network Formation for FSO Satellite Communication

Revital Marbel [1,*], Roi Yozevitch [2], Tal Grinshpoun [3] and Boaz Ben-Moshe [1]

[1] Ariel Cyber Innovation Center and Astrophysics, Geophysics & Space Science Research Center, Department of Computer Science, Ariel University, Ariel 4070000, Israel; benmo@g.ariel.ac.il
[2] Department of Computer Science, Holon Institute of Technology, Holon 5810201, Israel; roiyo@hit.ac.il
[3] Ariel Cyber Innovation Center, Department of Industrial Engineering & Management, Ariel University, Ariel 4070000, Israel; talgr@ariel.ac.il
* Correspondence: revitalm@ariel.ac.il

Abstract: Satellite network optimization is essential, particularly since the cost of manufacturing, launching and maintaining each satellite is significant. Moreover, classical communication optimization methods, such as Minimal Spanning Tree, cannot be applied directly in dynamic scenarios where the satellite constellation is constantly changing. Motivated by the rapid growth of the Star-Link constellation that, as of Q4 2021, consists of over 1600 operational LEO satellites with thousands more expected in the coming years, this paper focuses on the problem of constructing an optimal inter-satellite (laser) communication network. More formally, given a large set of LEO satellites, each equipped with a fixed number of laser links, we direct each laser module on each satellite such that the underlying laser network will be optimal with respect to a given objective function and communication demand. In this work, we present a novel heuristic to create an optimal dynamic optical network communication using an Ant Colony algorithm. This method takes into account both the time it takes to establish an optical link (acquisition time) and the bounded number of communication links, as each satellite has a fixed amount of optical communication modules installed. Based on a large number of simulations, we conclude that, although the underlying problem of bounded-degree-spanning-tree is NP-hard (even for static cases), the suggested ant-colony heuristic is able to compute cost-efficient solutions in semi-real-time.

Keywords: FSO; satellite laser communication; ant colony methodology; dynamic network optimization; satellite constellation interlink optimization

Citation: Marbel, R.; Yozevitch, R.; Grinshpoun, T.; Ben-Moshe, B. Dynamic Network Formation for FSO Satellite Communication. *Appl. Sci.* **2022**, *12*, 738. https://doi.org/10.3390/app12020738

Academic Editor: Fabrizio Granelli

Received: 12 December 2021
Accepted: 9 January 2022
Published: 12 January 2022

1. Introduction

Optimizing satellite communications is an important field in the new-space industry. Currently, most such communication is achieved by high-power radio frequency (RF) communication. In this research, we consider an optical laser-based communication network for a swarm of low-Earth-orbit (LEO) nano-satellites. The main motivation of creating such a network is to provide global internet coverage, i.e., to be able to communicate between any two points on Earth, especially in regions where terrestrial internet is rare or simply not available.

A global communication network requires two main types of communication:

- Inter-satellite or space-to-space communication, which enables the deployment of a space network comprised of LEO nano-satellites.
- Ground-space communication, which enables a link connecting between a ground station and a satellite; see Figure 1.

The focus of this paper is on space-to-space communication and particularly on the formation of an inter-satellite space network in view of two principal characteristics that are seldom addressed in the literature. The first characteristic is the dynamic properties of the network, in the sense that the topology of the network is constantly changing due to the orbit of each nano-satellite in the constellation.

The dynamic nature of the network is important in the context of optical communication. This is due to a required acquisition time between two satellites to establish an optical connection. Thus, our suggested network formation method aims to reduce the overall number of required modifications over time.

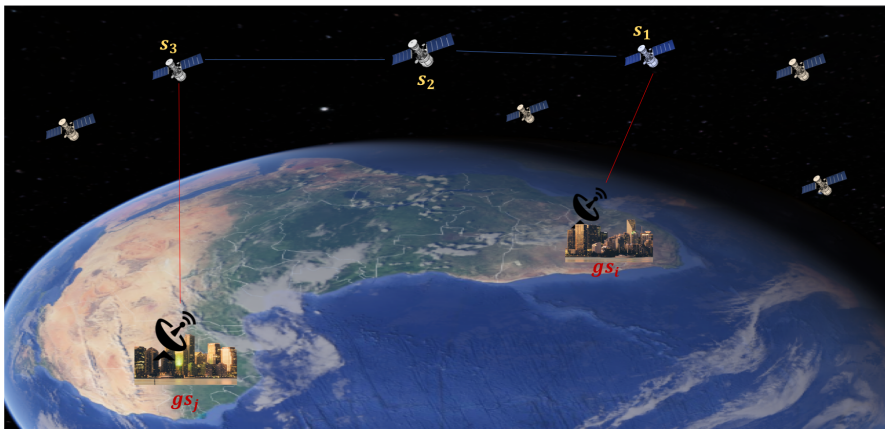


Figure 1. Ground station satellite communication illustration. This Figure demonstrates communication between 2 ground stations: gs_i, gs_j via 3 satellites: s_1, s_2, s_3.

The second characteristic addresses the number of peers a satellite can simultaneously communicate with at any given time. This characteristic stems from a dedicated optical module that is mounted on a satellite; a single optical module enables communication with a single peer satellite. Currently, this number is bounded to a small (constant) number of laser modules (commonly 3–5), which is significantly lower than the number of possible "neighboring" satellites. Thus, one can model the inter-satellite laser communication problem as a bounded degree spanning network (over dynamic graphs).

1.1. Related Works

Most research related to new space communication optimization are still addressing RF technology and often is led by commercial companies (e.g., Project Loon [1], NSLCOMM and Iridium NEXT [2]). Few new space research projects relate to laser-com: StarLink is a massive LEO constellation designed by SpaceX, and its main goal is to allow a global coverage, high-bandwidth and low latency satellite network. The Starlink backbone is based on laser-com between satellites [3], while the space-to-earth segment is based on RF communication [4].

FSO communication was suggested as an alternative to fiber optic networks and RF links [5,6] in terrestrial and urban scenarios. NASA has an optical communications program (see [7])—in particular, a lunar laser-com demonstration (see [8])—and the MIT Node Project focuses on laser-com for nano-satellites (see [9]). The goal of these projects is, ultimately, to create a network of satellites that will provide full coverage for the ground stations scattered around the Earth.

The communication between these satellites is planned to be optic-based at the end of the process. Laser communication is fast, secure and does not require a frequency license appose to RF communication. The field of FSO link optimization has been investigated in several aspects, including optimal modulation of maximizing the bandwidth [10,11], minimizing bit error rate [12] and security concerning optical jamming [13].

In addition, space has optimal conditions for optical communication because there is no atmosphere like on Earth. Nevertheless, such a network also has limitations; point-to-point optical communication requires accurate aiming that requires time. In addition, each satellite can have a limited number of optical communication components, and each component can be linked to one satellite at a time. These restrictions imply that the network has to change from time to time to provide full coverage.

Given the satellite infrastructure, the question arises, how to build a network that is optimal for transmitting data. This paper offers an algorithm for building such a network. The problem of networking a group of satellites that has limited components can be referred to as the Bounded-Degree Minimum Spanning Tree problem (BDMST).

In the Minimum Bounded-Degree Spanning Tree problem, the goal is to find a spanning tree in a given graph G that minimizes the graph cost and the graph degree to some constant n. Determine if there exists a spanning tree with maximum degree n is NP-Complete for $n = 2$ [14] since it captures the Hamilton-Path problem with cardinality n. The Minimum Bounded-Degree Spanning Tree problem has proved to be NP-hard; therefore, it appears unlikely that a polynomial time solution algorithm exists for the problem [15].

The problem was first defined by [16] who suggested a branch-and-bound algorithm to construct a Minimum Bounded-Degree Spanning Tree. This algorithm is the optimal solution for a tree with up to 100 vertices. The optimal solution for the instances with up to 200 vertices was later suggested by Gavish [17]. Instances with up to 500 vertices were considered in the study of Li et al. [18], which proposed a genetic algorithm that uses a direct tree representation.

In this paper, we consider the case of the Starlink satellite constellation that, as of Q4 2021, consists of over 1600 operational LEO satellites and thousands more are expected in coming years. Finding optimal routing schemes in a such massive constellation is extremely demanding. However, there are some efficient heuristics methods for finding sub-optimal solutions.

One of the heuristics, presented by Hanr and Wang [19], uses a genetic algorithm. They proposed a bi-objective genetic algorithm for d-MST problem with the violation degree as the second objective. Based on this bi-objective function, they created a new crossover, a local search scheme, a mutation and a new selection operator. Their algorithm converges with probability one to a globally optimal solution.

Another method, suggested by Souza and Martins [20], uses variable neighborhood search, where the search was guided to the shaking phase while using skewed functions. A hybrid algorithm based on tabu search and ant-colony optimization was proposed by Katagiri et al. [21] for the k-minimum-MST problem, which is a minimum cost subtree of formed with at least $k \leq n$ vertices. In this paper, we extend the ant-colony method to produce a cost-effective bounded degree spanning tree on a dynamic scenario (dynamic graph).

1.2. Our Contribution

The main novelty of this work lies in the semi-optimal Degree Bounded Spanning Tree (DBST) heuristic—this method provides a solution to the bounded degree spanning tree problem in a dynamic network that can also be used as a reasonable solution in real-time. To the best of our knowledge, this is the first work that uses the ant-colony methodology in order to optimize the interlinks of real-world large satellite constellations. Moreover, based on simulated experiments with real-world satellite data, both the correctness and the efficiency of the algorithm were verified.

1.3. Paper Structure

Following the above introduction, the next section is devoted to defining the problem of interest. We present both the problem's definition and also provide motivations. Section 3 covers generalizations of minimal spanning tree algorithms in dynamic scenarios, in other words: cases of dynamic graphs in which, due to the movement of the nodes, the properties of the edges between them change in time. Section 4 presents the min heuristic for solving the bounded degree spanning tree using ant-colony methodology. The presented heuristic is evaluated in Section 5, which presents a set of simulation results on different scales of constellations. Finally, Section 6 concludes the paper and discusses possible future work challenges.

2. Problem of Interest

We start by describing a number of key logical components that are required for formally stating the problem at hand:

- Communication link: two points (each point can represent a satellite) have a communication link if they are closer than a predefined minimum distance m_d. Denote the distance between two satellites s_i, s_j as $d(s_i, s_j)$, and we define a function: $LOS :\rightarrow \mathbb{R}^+$ so that: $LOS(s_i, s_j) = d(s_i, s_j)$ if $d < m_d$ and ∞ otherwise. LOS stands for Line-Of-Sight.
- Static Communication Graphs: a graph $G_c(t) =< V, E >$ where the vertices V represent the satellites. The edges E are defined from the LOS function, meaning, between two vertices in V there is an edge if there is a LOS between them. The weight function denotes as $W : E \rightarrow \mathbb{R}^+$ is defined so that $e = v_i, v_j$ and $W(e) = distance(v_i, v_j)$.
  Note that, by definition this graph is not necessarily connected. However, in this scenario, we assume a connected graph scenario. We will justify this assumption in the upcoming sections. Figure 2 demonstrates the communication graph computation.
- Dynamic Communication Graphs: a dynamic graph is a graph defined in a time interval, where the edges' weights change over time. The dynamic communication graph denotes as: $G_d(t) =< V, E, T >$ where the vertices V represent the satellites, and the edges E are defines by the LOS function in each point in time $t \in T$. The weight function denotes $W : \{E, T\} \rightarrow \mathbb{R}^+$ is defined so that $e = (v_i, v_j)$, and $W(e, t)$ is the distance between the vertices (v_i, v_j) in time t.

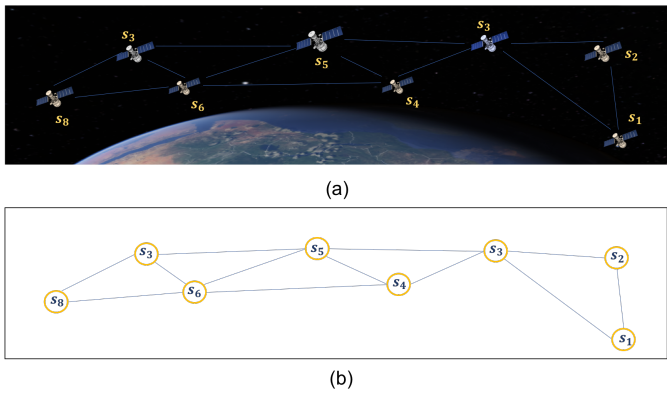


Figure 2. Communication graph illustration. (**a**), The satellites constellation and (**b**) the FSO communication graph created by such a constellation.

Equipped with the above terms, one can describe the problem's state. The problem can be defined as how to construct an *optimal* spanning graph that connects all the vertices in *every* time interval. By optimal, one means the smallest sub-network where every two points (satellites) can communicate either by a direct link or through other points (point A communicates with point B via point C). If one substitutes each vertex with a communication satellite and each edge with a communication link, the motivation is both clear and important.

A simpler version of this problem is well known in the literature as the Minimum Spanning Tree (or MST) problem [22]. As mentioned above, static graphs are graphs with static (unchanging) weights. A static MST problem (MST on static graphs) can be solved ($O(nlogn)$ by applying Kruskal's or Prim's algorithm [23].

However, since a satellite constellation is constantly changing, one faces a *dynamic* graph scenario where the edges' weights are changing over time, the MST is susceptible to changes. Theoretically, one can treat a dynamic scenario as a consecutive static-graph problems, each can be solved using classical methods. Alas, this approach raises two important ramifications that have not been addressed thus far.

First, changing the optimal network structure (due to satellites' movements) cannot be done instantly, especially not when using optic communications. Establishing an optic connection between two satellites is a delicate procedure that takes acquisition time. Thus, the dynamic scenario forces us to consider an alternative method for graphs spanning over time. It should be noted that the actual acquisition time of an inter-satellite laser link may vary and is hardware dependent, yet even few milliseconds may cause significant delays. For a detailed analysis of the expected delays and a comparison between fiber-optics and inter-satellite laser back-hauling, see [24,25] .

The second problem has to do with the maximum number of edges each vertex is allowed to have. An optic communication between two satellites is obtained by a dedicated optic module. One cannot have more than three to four such modules on a single satellite. Thus, it is impossible for such a satellite to establish a connection with more than three to four satellites (the number of modules). It is important to mention that inter-satellite laser communication is performed in the vacuum of space, and therefore the main parameter that effects such FSO links is the distance between every two neighboring satellites, which is represented by the weights of the edges.

Therefore, it is mandatory to limit (bound) the number of edges from each vertex. This modified MST version is called the Bounded-Degree MST, and this is a NP Hard problem [16]. We address the first problem (the dynamic scenario) in the next section and the bounded-degree constraint in the following section.

3. Minimum Spanning Tree on Dynamic Graphs (DMST)

As explained above, an MST on a dynamic graph cannot be solved as different $MSTs$ on consecutive static graphs. Let us first define what a dynamic graph MST means and how to define it.

We start with a discrete scenario, and then we generalize to a continuous scenario.

3.1. Discrete Dynamic Graphs MST

A discrete dynamic graph at time interval $T = [0, t]$ is defined as an orderly set of graphs sampled at different timestamps between 0 and T. We assume no topological changes, i.e, only the edges' weights are changing. The minimal spanning graph during T will be defined as the MST. Put it differently, from all the edges in the complete graph, only the ones whose sum is minimal over time will be chosen.

Figure 3 shows the difference in the MST on a simulated star-link graph in four different states in time. Between the states, there is only one second. This figure illustrates the problem of computing a bounded degree MST in short time intervals.

Algorithm 1 describes the optimal spanning tree for dynamic communication graph process.

Algorithm 1: Optimal Dynamic Communication Tree Algorithm

Input: Set of z communication graphs: $G = g_1 =< N, E_1 >, \ldots g_z =< N, E_z >$
Result: An optimal spanning tree $T =< N, E >$

```
let G' =< N, ∅ > be an empty graph
for g in G do
    for e in E(g) do
        w(G'(e))=w(g(e))+w(e)
T = ST(G') (T is the minimum spanning tree of G') return T
```

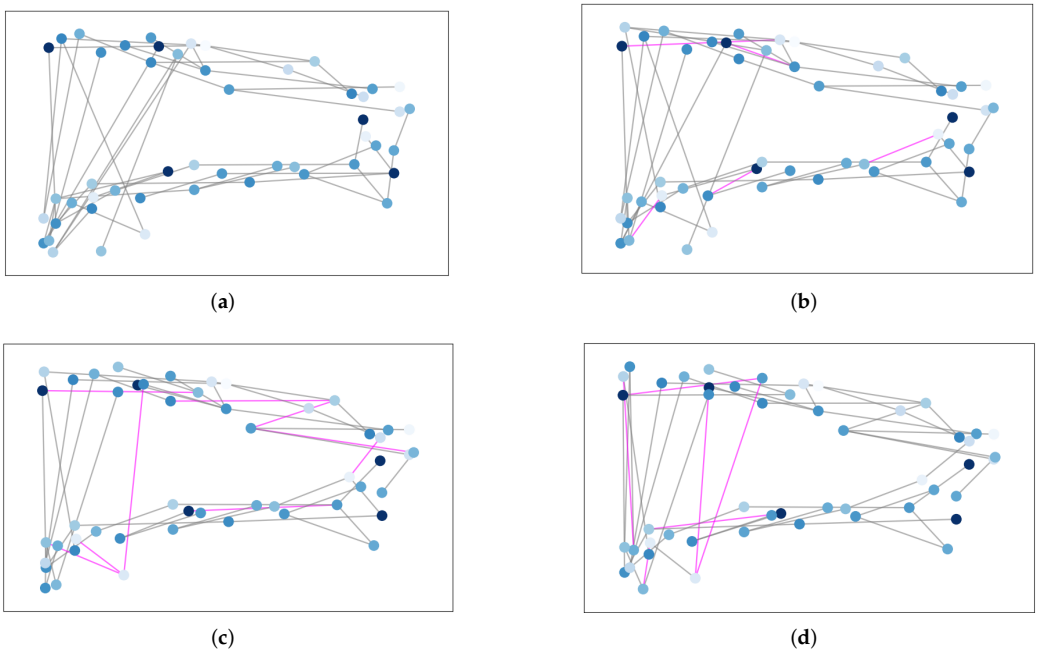


Figure 3. Dynamic MST. A 50-vertices star-link graph in four states one second apart. The magenta edges are the new edges that were added from the previous states. The vertices are colored differently so that the graph motion is easier to track. (**a**) MST in time 0. (**b**) MST in time 1. (**c**) MST in time 2. (**d**) MST in time 3.

Figure 4 illustrates the algorithm by an example. Table 1 illustrates how the algorithm constructed the minimal spanning tree for that example.

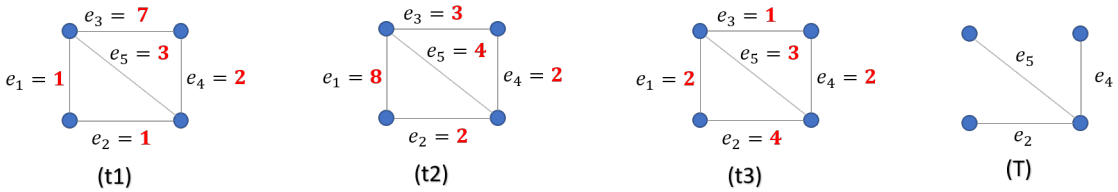


Figure 4. Algorithm example: communication graph constellation at three points in time (t1–t3) and the optimal spanning tree (T).

Table 1. MST on a dynamic graph algorithm example.

Edges	t1	t2	t3	T
e_1	1	8	2	11
e_2	1	2	4	7
e_3	7	3	1	11
e_4	2	2	2	6
e_5	3	4	3	10

This algorithm is based on Kruskal's algorithm for finding a spanning tree in a static graph. The Kruskal's algorithm method for finding a minimum spanning tree is to start with an empty graph (only vertices and no edges) and to gradually add low cost edges until the graph is fully connected. Herein, we use that method but on an accumulating graph that is the sum of the edges costs through all the graphs. This algorithm is an extension to Kruskal's algorithm for a dynamic MST, and therefore its correctness relays on the dynamic MST definition and the correctness of Kruskal's algorithm.

3.2. MST for Continuous Dynamic Graph

After presenting algorithm that computes the minimal spanning tree in dynamic graphs where the function on the edges is discrete, this subsection will discuss the continuous form of this problem. One method of handling the continuous mode is to arbitrarily choose number of points in time and use the algorithm on these points. Figure 5, and Table 2 shows a case where this method will not provide the optimal solution.

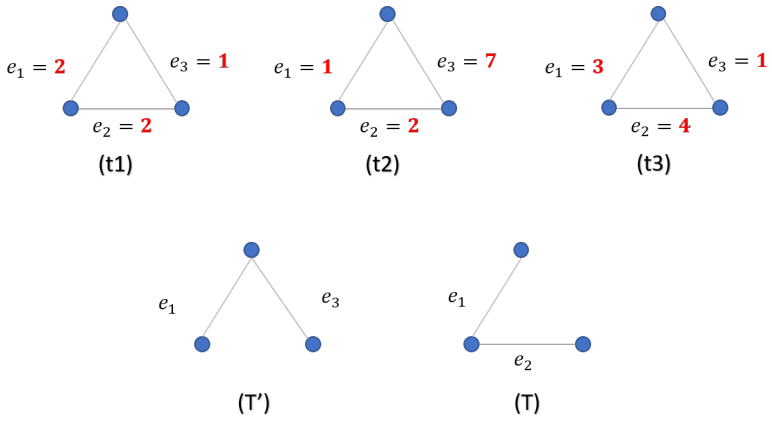


Figure 5. Algorithm example: communication graph constellation at three points in time (t1–t3). The optimal tree (T) and the algorithm result (T′) on trees (t1) and (t2), which is not optimal.

Table 2. MST algorithm example.

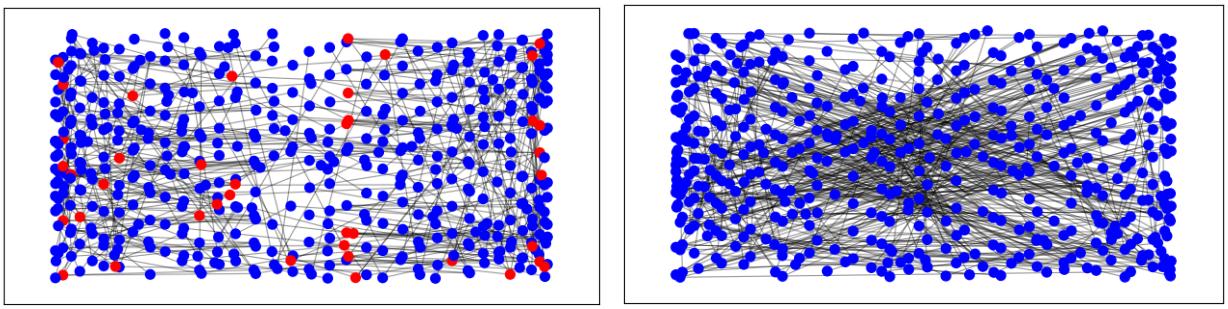
Edges	t1	t2	t3	T
e_1	2	1	3	6
e_2	2	2	4	8
e_3	1	7	1	9

Observation: the functions that represent the edges weight on dynamic graphs when dealing with satellite constellations are continuous functions. Since the weight of this edges is the distance between two satellites and due to the fact that the satellites moves in continuous orbit. We can rely on this observation to argue that, for dynamic graphs, we can provide a continuous solution when computing the integral over the edges in time instead of summing the edges weight in the discrete version.

4. The Degree Bounded Spanning Tree (DBST) Heuristic

As explained in Section 3.2, the spanning of a dynamic FSO-based network cannot be fully achieved using MST, since each satellite (vertex) has a limited amount of communication components it can carry (usually three or four). Thus, the number of links (or edges) for each vertex is bounded. This problem is referred to in the literature as "the bounded degree minimum spanning tree" [26]. This section presents a heuristic method for computing a cost-effective bounded spanning tree on dynamic graphs. Note that the bounded degree minimum spanning tree problem is NP-Hard even for static graphs. This motivated the offered heuristic attempts to provide a low cost solution in a reasonable time.

Figure 6 shows an example of why an algorithm for finding unbounded *MST* will yield insufficient results that do not match the optimization we are looking for. This figure describes two spanning graphs of the same satellite-based network of 500 vertices. Figure 6a shows the minimum spanning tree with no degree limit (calculated by a Kruskal algorithm); the vertices painted in red are vertices with a rank higher than 3. Figure 6b depicts a (non-minimal) spanning graph of the same graph with all vertices less than 3.



(**a**) (**b**)

Figure 6. (**a**) MST of a 500 vertices graph. The red notes degree greater than three. (**b**) Three bounded degree spanning tree of the graph presented in (**a**).

Previous works have tackled this problem in different methods [27–29]. One of the methods utilizes an Ant-Colony (AC) algorithm [30].

AC algorithms are inspired from ants' behavior in their colony and the way they are seeking for food in their vicinity. AC algorithms are mainly used to find the shortest path on the traveling salesman problem (another NP-hard problem). Each ant solves the search problem by walking a certain path and leaves behind a pheromones trail to guide the next ant to come. The next ant will prefer to choose a way that has a strong pheromones scent, an indication that other ants already used it. After several such cycles, all the ants manage to find the shortest way to the food.

The intuition for choosing the AC method to build such a communication network where the vertices are connected via minimum weight is as follows: a standard MST algorithm chooses the low weighted edges until reaching a tree while avoiding circles (using the Kruskal algorithm). This algorithm provides an MST with some vertices having higher degree than allowed. This method can be adopted to bounded degree trees by 'replacing' edges that violate the degree restriction and using a bypass to connect the vertex to the rest of the network (via other vertices).

However, choosing such a path is hard and relies on the question: what is the shortest path between two connected component on graph. With the AC method, the ants mark the best edges to use for minimizing the path between the graph vertices. Building the tree using the pheromone level could yield a graph that connects the vertices using optimal paths.

To the best of our knowledge, all the previous methods that tackled this problem were restricted to static graphs only. The novelty of this work is the adaptation of the AC method for dynamic bounded-degree spanning trees. The rest of this section describes both static and dynamic bounded-degree graphs using the AC algorithm.

4.1. AC Algorithm for Finding Low Cost Degree Bound Spanning Tree

The algorithm is divided into the following stages:

1. Initialization. The ants are randomly assigned to the graph vertices.
2. Search. The ants "move" on the graph when they prefer high cost edges in high probability. A visited edge will be marked to update the pheromone levels in the next step.
3. Graph construction. Creating a spanning graph using the pheromone level on the edges while avoiding the degree-bound restriction.
4. Quality check. If the tree's weight is lower than the tree we calculated earlier, it will be the new candidate. If not: Rearrange the pheromone level to avoid a local minimum situation (Evaporate pheromones).

Figure 7 presents the Ant-Colony algorithm for dynamic graph flows.

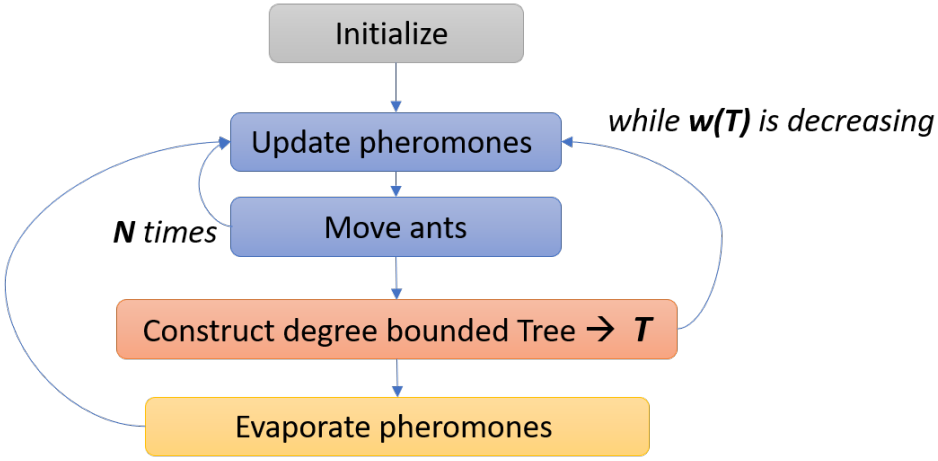


Figure 7. The Ant-Colony algorithm flow.

The main method of this algorithm is presented in Algorithm 2. The method inputs are the graph, the edge weights and the number of steps, which is the number of attempts to build a spanning tree. The main method first assigns ants to randomly chosen vertices in the graph, initiates the pheromone level for each edge and then loops over the number of steps where each step moves the ants along the graph and builds a tree described in Algorithms 3 and 4.

Algorithm 2: AC algorithm for low cost degree-bound spanning tree: Main method.

Input:
- communication graph G
- a weight function for each edge w
- maximum degree: λ
- number of steps: n and number of ants a

Result: low-cost degree-bound spanning tree T

```
let T =< N, ∅ > be an empty graph
assign a ants to random vertex in G
init pheromones on G edges
for n-steps do
    T=Move Ants and Build Spanning Tree Algorithm 3 (G,w,T, λ)
return T
```

Algorithm 3 describe the ants' movement on the graph and the pheromones update. For a predefined number t times, the ants move on the graph, and the ants move from one vertex to another favoring the low cost edge (with some probability) to travel along. On the edge chosen, the pheromone level will be updated using this formula:

$$Ph = \epsilon \times Ph_{old} + (1 - \epsilon)\frac{1}{cost} \quad (1)$$

where Ph_{old} is the previous pheromone level, $cost$ is the edge cost, and ϵ is a predefined parameter used to define the weight of the new calculation on the pheromone level against the old pheromone level.

After moving the ants a few times, the algorithm attempts to construct a tree based on the pheromone level on the edges; the construction method is also based on Kruskal's algorithm or contracting a minimum spanning tree but taking into account the degree bound.

Algorithm 3: Move ants and build spanning tree algorithm.

Input:
- communication graph G
- a weight function for each edge w
- a Tree T
- maximum degree λ

Result: low-cost degree-bound spanning tree T

```
let C = w(T)
while w(T) is decreasing do
    for t times do
        move ants(G)
        update pheromones(G)
    T=construct tree(G, λ) (using Algorithm 4)
    if w(G') ≤ c then
        c = w(G')
        T = G'
evaporate pheromones(G)
return T
```

4.2. Local Minimum

The Ant Colony algorithm performs well against NP-hard problems. However, it has an inherent disadvantage of falling in local minimum, which is very likely in the addressed problem of satellite inter-links given the massive number of vertices and iterations. This is why the evaporation process is commonly utilized in AC algorithms. The probabilistic nature of this process can overcome those problems [31]. Our implementation of the evaporation process is elaborated in Algorithm 3.

Algorithm 4 describes the tree construction process after each loop.

Algorithm 4: Tree construction algorithm.

Input:
- dynamic graph G and the pheromone level on the edges
- degree-bound λ

Result: low-cost degree-bound spanning tree T

```
let T=<N,∅> be an empty graph
SE=sort G edges by pheromone level
while T is not connected do
    let e={ i,j } be the first edge on SE
    if ( adding e does not form a circle in T) and (after adding e i and j degree is ≤ λ) then
        T=T+e
        remove e
return T
```

4.3. Optimizing the Next Step: Pheromones Transfer

After describing the AC method used for finding a low cost bounded degree spanning tree a static graph, this subsection describes how an extension can provide a fair solution on a dynamic graph as well. The algorithm described in this section uses the observations from the low-cost spanning tree problem on dynamic graphs and the algorithm presented in Algorithm 2. The method will be performed continuously for each time interval of the dynamic graph. At the starting point (first time interval), the algorithm will perform the method for static graph but, for the next time intervals, the algorithm will use the

previously calculated graph pheromones, for a prediction on edge at the time interval. Algorithm 5 describes the continuous process.

Algorithm 5: AC algorithm for low-cost degree-bound spanning tree: continuous method.

Input:
1. dynamic graph G=<V,E>
2. a weight function for each edge w
3. maximum degree:λ
4. previous state graph: prev
5. Pheromones percent to copy P

Result: low-cost degree-bound spanning tree T

1 let T=<N,∅> be an empty graph
2 E=chose P percent of the edges in prev to reassign pheromones
3 copy pheromones in prev(E) in to G(E)
4 T= Algorithm 3 (G,w,T, λ) from line 5.
5 **return** T

The continuous method will be performed on each time interval. Each time the method obtains the previous state graph *prev*, which holds the latest calculated pheromones, the method will copy some of the pheromones to the current graph in order to speed up the calculation. The top 50% of the pheromones are chosen for this process. Since the graph is continuous, it is a reasonable assumption that optimal edges will keep their score while sub-optimal edges will be replaced.

We rely on the observation that, between time intervals, the majority if the edges weights will not change significantly, considering that the satellite movement is continuous. However, the method does not copy all the pheromones from the previous state in order to avoid a local minimum. A predefined parameter P represents the percentage of pheromones to copy. The rest of the edge pheromones will be initiated the same as in the first state. After copying the pheromones, the method will perform the main loop described in Algorithm 2.

5. Results

In order to test the proposed method on real data, we built a simulator that used real-time satellite motion information. We created a dynamic graph from this data and tested our algorithm (nDBST) with respect to the run time, convergence and final graph cost. This section describes both the simulator itself and the results on static and dynamic graphs.

5.1. Simulator

The real-time data was obtained from the Starlink satellite network. Since Starlink satellite locations are available online (and can be even computed for two weeks in advance), it was a perfect candidate to verify the algorithm's correctness. A series of dynamic graphs were created and embedded using two open source Python libraries:

- Networkx [32]. This library was developed for the study of complex networks and has method implementations, such as computing MST, check connectivity and computing the connected components of graph algorithms.
- Ephen [33]. Used for performing high-precision astronomy computations.

The simulation can be conceptually divided to two steps: pre-process and method testing. The pre-process step retrieves the Starlink satellite locations using the Ephem library, computes the distance between the satellites, and assigns each edge with the respective cost as described in Section 2. For the method testing step, we implemented the low-cost bounded degree method described in Algorithm 2 and tested its performance on these graphs. The code used for testing the method can be found at https://github.com/RevitalMarbel/AC_for_dnbdst (accessed on 24 October 2021).

The next subsection presents the AC method results on typical dynamic network.

5.2. Experiment Results

Graphs of four sizes were created from the Starlink satellites (randomly choosing satellites as vertices). The satellite locations were used to calculate the edge cost in each time interval. The AC algorithm was performed on each graph for several steps measuring the convergence rate after each step.

As expected, the final cost of the spanning graph is reduced as the algorithm performs more steps. Figures 8 and 9 show the AC method results on these graphs. For each time interval (steps), the total cost of the constructed graph using the tree construction method is presented. Finally, the difference from the MST is normalized to the average edge cost and the number of vertices in the graph in order to measure the difference in the costs with respect to an average edge cost. The result in each step is the graph cost, computed by the sum of all edges. For the sake of comparison, we normalized the graph cost by measuring the distance from the minimum spanning tree cost using this formula:

$$norm = \frac{cost - mst}{average}$$

where: *cost*: is the constructed graph result, *average* is the average edge cost, and *mst* is the minimum spanning tree cost of the graph.

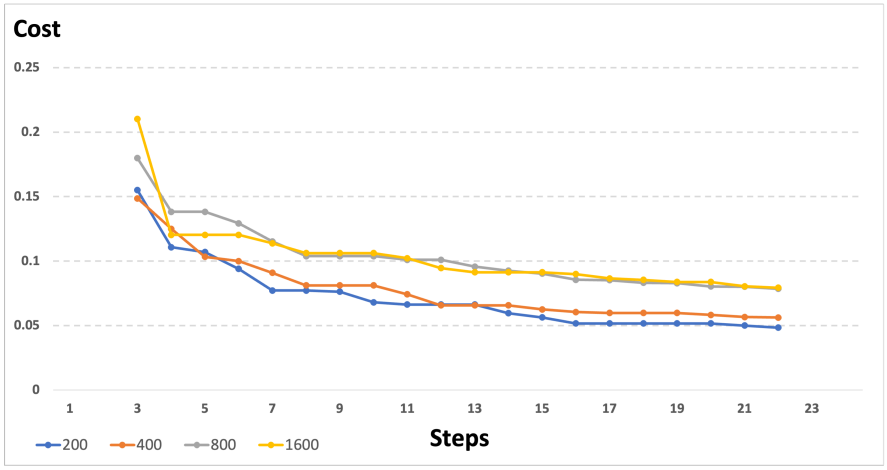


Figure 8. AC for the low-cost bounded degree spanning tree algorithm after 20 steps.

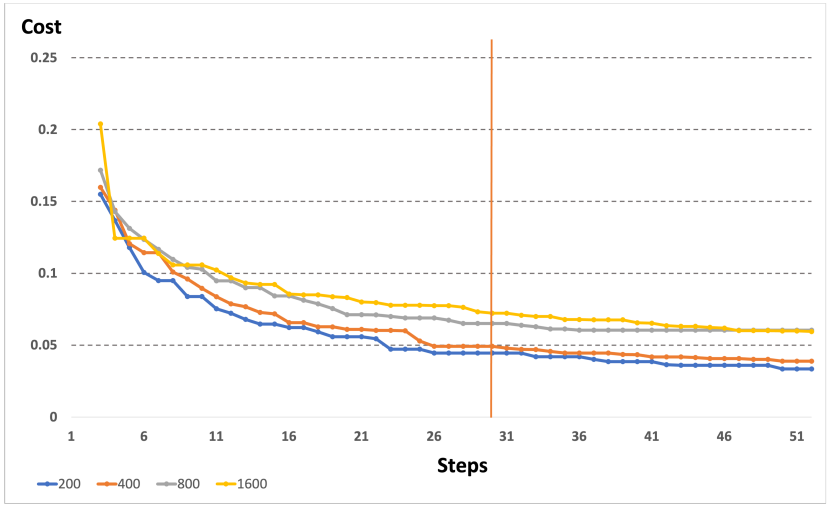


Figure 9. AC for the low-cost bounded degree spanning tree algorithm after 50 steps.

The experiment included graphs in different sizes: 200, 400, 800 and 1600 vertices. On each of these graphs, we allocated ants on half of the edges randomly. We ran the static method for 20 steps and printed the cost reduction of T. We found that 20 steps, as the next table shows, was usually sufficient to reach a result that is close to the minimum (MST). In order to test the method abilities to reduce the cost over time, we extended the run-time by performing the method for 50 steps. The result show that, after 30 steps, on average, the extra cost reduction was insignificant (Figure 9).

Finally, we tested the method of Algorithm 5 for continuous dynamic low-cost degree-bound spanning trees. The main goal was to prove that the method converged to a solution faster after pheromone transfer compared with performing that static method on each time interval. We performed the following experiment: on 200, 400, 800 and 1600 sized graphs, we computed five consecutive communication graphs (representing five time-intervals). On these graphs, we performed two experiments.

The first experiment was attempting the static method on each graph, the same as in Figure 8, for 10 steps, and, in the second method, we performed the continuous method with pheromone transfer. We computed the bounded spanning tree cost after each step and presented it in a graph. Figure 10 shows the different method results. The graph shows that the continuous method improved the cost reduction rate significantly.

One can see that, if the algorithm starts with a random set of pheromones, the initial convergence will be very rapid (since the initial guess is random). This phenomenon is depicted in the rapid decay of the orange line in the beginning of each sequence. On the other hand, when the computation starts with previously computed pheromones, since those pheromones are already close to optimal, the convergence will be much slower as illustrated by the blue line.

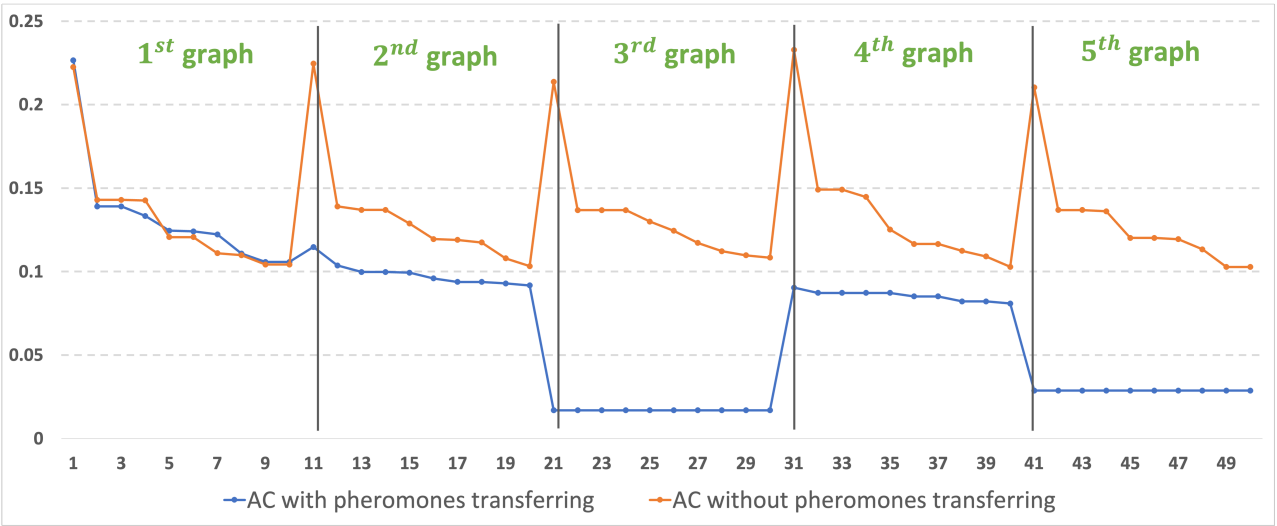


Figure 10. AC for low-cost continuous bounded degree spanning tree algorithm for an 800-vertex graph.

The system run-time for each graph size is given by Table 3. The table shows the average time of each step on different graphs, each step includes moving the ants 150 times, updating the graph data and constructing the spanning tree.

Table 3. AC algorithm run-time result.

Size (number of vertices)	200	400	800	1600
Run-time (seconds)	1.3	5.2	22.71	100.23

In order to decide what is the optimal step nail to take to get to the complexity threshold, one has to consider few parameters: (i) The graph size (the number of vertices).

(ii) The number of edges. (iii) The rate of topological change of the graph over time. The FSO graph is basically a geometric (visibility) graph in which two nodes may establish a link if the distance between them is below some threshold. Therefore, the graph's average degree increases proportionally to the number of nodes.

Therefore, in relatively small graphs (up to 400 nodes) due to a relatively small number of optional routing solutions, the suggested method may get caught in a local minimum, While in dense graphs (e.g., 800 and 1600 nodes), the suggested method seems to be able to "route out" of local minimums. Figure 9 and Table 3 show a clear trade-off between the times where it is possible to let a single graph converge and the rate at which the dynamic graph is cut.

6. Conclusions

This study proposed a method for calculating a cost-effective bounded degree tree for a dynamic network. The motivation for the study was to suggest a method for the fast calculation of intelligent networks based on satellite interlink optical communications, such as SpaceX's massive Starlink constellation.

This paper offers some essential insights into computing such a network: (i) Frequent shifting of the network edges causes major topological changes in the underlying FSO communication graph. Therefore there is a need for a dynamic heuristics applicable of computing runtime solutions for online input scenarios. (ii) In optical communication-based networks, establishing a new link takes time. Hence, an existing link should last for a reasonable duration before creating a new one. (iii) The satellite structure limits the number of concurrent neighbors that each node can be connected to.

Given these observations, a method for searching for cost-effective bounded degree spanning trees is presented. Considering that the problem of finding the minimum bounded spanning tree is an NP-hard problem, a heuristic method for a cost-effective solution on a static graph was designed. This method searches for such a tree in a reasonable run-time. The main novelty of this work lies in the DBST method—this method provides a solution to the bounded degree spanning tree problem in a dynamic network that can also be used as a reasonable solution in real-time. Simulation experiments with real-world satellite data verify both the correctness and the efficiency of the algorithm.

In the coming years, the size and complexity of LEO constellations (such as Starlink) is expected to grow dramatically, supporting millions of communication links [34]. Moreover, the overgrowing demand for reduced latency communication [35] raises several QoS (Quality of Service) routing problems, as some communication links are extremely sensitive to latency, while others can handle a relatively considerable delays.

A natural future work may include generalizing the problem to be able to adjust to a dynamic communication demand. In this generalization, a large set of communication demands are presented as an offline (or online) ground-station to ground-station (peer2peer) communication demands. The goal was to find the spanning graph that minimizes the weighted cost (or delay) over all the communication demands.

In order to validate the performance of such a complicated (real-world) problem, there is a need to construct a detailed benchmark on which both online and offline methods can be evaluated. Having both the benchmark and the generalized methods, one should be able to compare "Starlink-like" solutions to existing fiber-optic solutions—which appears to be a major optimization problem related to 5G global networks.

Author Contributions: Conceptualization, R.M. and B.B.-M.; methodology, R.M., R.Y., T.G. and B.B.-M.; software, R.M.; validation, R.Y., T.G. and B.B.-M.; formal analysis, T.G.; investigation, R.M., R.Y., T.G. and B.B.-M.; writing—original draft preparation, R.M., R.Y., T.G. and B.B.-M.; writing—review and editing, R.M., T.G. and B.B.-M.; visualization, R.M.; supervision, T.G. and B.B.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data supporting the reported results can be found at https://github.com/RevitalMarbel/AC_for_dnbdst (accessed on 24 October 2021)

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Katikala, S. Google project loon. *Insight Rivier Acad. J.* **2014**, *10*, 1–6.
2. Gupta, O.; Fish, C. *Iridium NEXT: A Global Access for Your Sensor Needs*; AGU Fall Meeting Abstracts; American Geophysical Union: New Orleans, LA, USA, 2010.
3. Toyoshima, M. Recent Trends in Space Laser Communications for Small Satellites and Constellations. In Proceedings of the 2019 IEEE International Conference on Space Optical Systems and Applications (ICSOS), Portland, OR, USA, 14–16 October 2019; pp. 1–5.
4. Tang, Z.; Li, S.; Deng, W.; Wang, Y.; Yu, W. Optimization of Satellite-Ground Coverage for Space-Ground Integrated Networks Based on Discrete Global Grids. In Proceedings of the International Conference on Space Information Network, Wuzhen, China, 19–20 September 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 132–144.
5. Marbel, R.; Ben-Moshe, B.; Grinshpoun, T. Urban Free-Space Optical Network Optimization. *Appl. Sci.* **2020**, *10*, 7872.
6. Wang, Y.; Wang, P.; Liu, X.; Cao, T. On the performance of dual-hop mixed RF/FSO wireless communication system in urban area over aggregated exponentiated Weibull fading channels with pointing errors. *Opt. Commun.* **2018**, *410*, 609–616.
7. Cornwell, D.M. NASA's optical communications program for 2017 and beyond. In Proceedings of the 2017 IEEE International Conference on Space Optical Systems and Applications (ICSOS), Naha, Japan, 14–16 November 2017; pp. 10–14.
8. Boroson, D.M.; Scozzafava, J.J.; Murphy, D.V.; Robinson, B.S.; Lincoln, M. The lunar laser communications demonstration (LLCD). In Proceedings of the 2009 Third IEEE International Conference on Space Mission Challenges for Information Technology, Pasadena, CA, USA, 19–23 July 2009; pp. 23–28.
9. Nguyen, T.N.T. Laser Beacon Tracking for Free-Space Optical Communication on Small-Satellite Platforms in Low-Earth Orbit. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2015.
10. Paul, P.; Ghosh, S.; Bhatnagar, M.R. Abating Jamming in Free Space Optical Systems–A Game-theoretic Solution. *IEEE Trans. Commun.* **2021**, *69*, 8375–8387.
11. Chauhan, I.; Paul, P.; Bhatnagar, M.R.; Nebhen, J. Performance of optical space shift keying under jamming. *Appl. Opt.* **2021**, *60*, 1856–1863.
12. Bhatnagar, M.R.; Nebhen, J. Feedforward-based free-space optical communication. *Appl. Opt.* **2021**, *60*, 3155–3161. [PubMed]
13. Sharda, P.; Jaiswal, A.; Bhatnagar, M.R.; Garg, A.; Song, L. Clustering Based Diversity Improving Transmit Laser Selection Schemes Using Quantized Feedback for FSO System. *IEEE Trans. Veh. Technol.* **2021**, *70*, 6855–6868.
14. Ravi, R.; Sundaram, R.; Marathe, M.V.; Rosenkrantz, D.J.; Ravi, S.S. Spanning trees—Short or small. *SIAM J. Discret. Math.* **1996**, *9*, 178–200.
15. Hartmanis, J. Computers and intractability: A guide to the theory of NP-completeness (michael r. garey and david s. johnson). *Siam Rev.* **1982**, *24*, 90.
16. Narula, S.C.; Ho, C.A. Degree-constrained minimum spanning tree. *Comput. Oper. Res.* **1980**, *7*, 239–249.
17. Gavish, B. Topological design of centralized computer networks—Formulations and algorithms. *Networks* **1982**, *12*, 355–377.
18. Li, Y. *An Effective Implementation of a GA Using a Direct Tree Representation for Constrained Minimum Spanning Tree Problems*; Technical Report, Technical Report 2000–2015. LaRIA; Universit e de Picardie Jules Verne: Amiens, France, 2000.
19. Hanr, L.; Wang, Y. A novel genetic algorithm for degree-constrained minimum spanning tree problem. *Int. J. Comput. Sci. Netw. Secur.* **2006**, *6*, 50–57.
20. de Souza, M.C.; Martins, P. Skewed VNS enclosing second order algorithm for the degree constrained minimum spanning tree problem. *Eur. J. Oper. Res.* **2008**, *191*, 677–690.
21. Katagiri, H.; Hayashida, T.; Nishizaki, I.; Guo, Q. A hybrid algorithm based on tabu search and ant colony optimization for k-minimum spanning tree problems. *Expert Syst. Appl.* **2012**, *39*, 5681–5686.
22. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; MIT Press: Cambridge, MA, USA, 2009.
23. Prim, R.C. Shortest connection networks and some generalizations. *Bell Syst. Tech. J.* **1957**, *36*, 1389–1401.
24. Handley, M. Delay is not an option: Low latency routing in space. In Proceedings of the 17th ACM Workshop on Hot Topics in Networks, Redmond, WA, USA, 15–16 November 2018; pp. 85–91.
25. Hu, J.; Cai, L.; Zhao, C.; Pan, J. Directed percolation routing for ultra-reliable and low-latency services in low earth orbit (LEO) satellite networks. In Proceedings of the 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), Virtual Conference, 8 November–16 December 2020; pp. 1–6.
26. Goemans, M.X. Minimum bounded degree spanning trees. In Proceedings of the 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), Berkeley, CA, USA, 21–24 October 2006; pp. 273–282.
27. Singh, M.; Lau, L.C. Approximating minimum bounded degree spanning trees to within one of optimal. *J. ACM* **2015**, *62*, 1–19.
28. Fekete, S.P.; Khuller, S.; Klemmstein, M.; Raghavachari, B.; Young, N. A network-flow technique for finding low-weight bounded-degree spanning trees. *J. Algorithms* **1997**, *24*, 310–324.

29. Imtiaz, M.N.; Ali, M.A. An Improved Ant-Based Algorithm for Minimum Degree Spanning Tree Problems. *IOSR J. Comput. Eng.* **2012**, *6*, 6–10. [CrossRef]
30. Sundar, S.; Singh, A.; Rossi, A. New heuristics for two bounded-degree spanning tree problems. *Inf. Sci.* **2012**, *195*, 226–240.
31. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39.
32. Hagberg, A.; Pieter Swart, D.S. Nentworkx. 2014. Available online: https://networkx.org/ (accessed on 24 October 2021).
33. Rhodes, B.C. PyEphem Astronomy Library. 2020. Available online: https://rhodesmill.org/pyephem/ (accessed on 24 October 2021).
34. Chen, Q.; Giambene, G.; Yang, L.; Fan, C.; Chen, X. Analysis of Inter-Satellite Link Paths for LEO Mega-Constellation Networks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 2743–2755.
35. Lai, Z.; Li, H.; Li, J. StarPerf: Characterizing Network Performance for Emerging Mega-Constellations. In Proceedings of the 2020 IEEE 28th International Conference on Network Protocols (ICNP), Madrid, Spain, 13–16 October 2020; pp. 1–11.