IPAM Research in Industrial Projects for Students (RIPS)

Optimizing Network Topologies

Final Report Prepared for
Aerospace

August 21, 2008

RIPS Team

Jacob Hughes (project manager) Kevin Ventullo
Eric Webb Gary Wilkins

Faculty Mentors

Dr. Maria D'Orsogna
Dr. Tom Chou

Sponsoring Mentors

Dr. Mark Coodey
Don Lanzinger

This project was jointly sponsored by The Aerospace Corporation and NSF Grant DMS-0439872.

Abstract

We have developed an efficient algorithm capable of generating optimal satellite networks given a set of constraints on satellite communication links. An optimal or close-to-optimal network has the traits of high connectivity, fast communication times (i.e. low latency), and high redundancy. The inputs for this algorithm are time-based lines of sight for satellite constellations and specific cross link constraints. The outputs are optimized network configurations for each time. There are two phases to the algorithm. In the first phase, we consider the satellites fixed in space, and use a genetic algorithm to find several optimal configurations. After doing this for each discrete time, the second phase then determines which configurations to use at each time while taking into account the dynamic nature of the satellite constellation.

Contents

List of Figures

List of Tables

Acknowledgments

We would like to thank Dr. Mark Coodey and Don Lanzinger at the Aerospace corporation for their support and guidance throughout this project. The project was completed during the 2008 Research in Industrial Projects program at the Institute for Pure and Applied Mathematics at the University of California, Los Angeles. We are also grateful to Analytical Graphics, Inc. for giving us access to their Satellite Tool Kit software package for our project. Finally, we would like to thank both Aerospace Corporation and the National Science Foundation for their joint funding of the project.

Chapter 1

Introduction

Since 1960 The Aerospace Corporation has operated a federally funded research and development center in support of the space program. Federally funded research and development centers, or FFRDCs, are unique independent nonprofit entities sponsored and funded by the U.S. government to meet specific long-term technical needs of varying government agencies that cannot be met by any other single organization.

The Aerospace Corporation's FFRDC is sponsored by the United States Air Force, and provides objective technical analysis and assessments for space programs that serve the national interest. As the FFRDC for space national security, Aerospace supports long-term planning and the immediate needs of our nation's military and reconnaissance space programs. In addition, the Aerospace FFRDC provides scientific and engineering support for launch operations, space networks, and related ground systems. This end-to-end involvement reduces development risks and costs, and enhances the probability of mission success.

The Aerospace Corporation is involved in developing satellite networks. For satellites to be of use to us on Earth, they must be able to effectively transmit information to stationary ground stations. However, satellites are not always able to directly transmit information to ground stations, due to Earth-blockage, weather issues, nuclear scintillation, and jamming. When direct communication between a satellite and ground station is not available, it may be possible to relay information through other satellites to the ground station. For this reason, it is necessary to develop an effective system of cross links that connect up satellite networks. Our project is to develop an algorithm to optimize satellite connections.

We are given data describing, at each minute, all possible pairs of satellites that can see each other. We must determine how best to choose which possible connections to utilize, subject to a set of connection constraints. Satellites are not equipped with many cross links, due to the physical and financial constraints of satellite development. So even if a satellite is in range of a number of other satellites, it may be the case that it is only able to connect to a subset of those in range, due to its limited number of cross links.

Our project can be divided into two phases. In the first, we solve for the optimal network configuration for each minute, not taking into account previous or future positions of the satellites.

In the second phase of our project, we consider a network over time. We can not simply implement the best configuration at every time step independently, since it may be the case that transitioning from one good configuration to another results in poor network connectivity in the transition. The additional challenge for the second phase is to optimize the network over a long period of time, including the transitional phases as well.

The test input for the algorithm that implements our solution will be a day's worth of possible connection information for three distinct satellite networks designed by The Aerospace Corporation. These sample networks will include a Medium Earth Orbit (MEO) constellation, and a "larger" and a "smaller" Low Earth Orbit (LEO) constellation.

Chapter 2

Background

2.1 Mathematical Description of the Problem

The problem of finding an optimal series of connections among satellites and a ground station is best approached by manipulating *graph* structures. A graph G is defined to be a set of *nodes*, representing satellites, along with a set of pairs of these nodes, known as *edges* or *links*, representing a line of sight between the associated satellites, $E(G)$. A graph can be visually represented as dots representing nodes, and lines connecting pairs of dots, that represent edges. If two nodes are connected by an edge they are said to be *neighbors.* The number of neighbors a node has is called the *degree* of the node. In our problem, each satellite has a maximum allowable degree, corresponding to the number of simultaneous active links possible. This maximum degree is based on the cross link constraints for each satellite. A graph is said to be *connected* if it is possible to reach any node from any other node by traveling along edges. A graph will be called *redundant* if any single node can be removed (along with its adjacent edges) without disconnecting any part of the graph. A *subgraph* of a graph G is a graph H such that every node of H is a node of G, and every edge of H is an edge of G. A *component* is a subgraph where each node in the component is connected with all other other nodes in the component via edges.

At each point in time, we can construct the *visibility graph*, in which nodes represent all satellites and base stations, and edges represent all lines of sight among the nodes. The edges in this graph have real-valued weights which represent the distance between the two nodes. From here, our task is to find a feasible *connection graph*, that is, a subgraph of the visibility graph where the edges now represent an actual link between two satellites/stations. As discussed earlier, the connection graph is limited in that the degree k of any node must obey $k \leq k_{max}$, where k_{max} is a parameter describing the number of cross links available to that node. Furthermore, there may be multiple types of cross links, and a single satellite will utilize a mixture of types of connections.

In general, enumeration of graph structures is impractical. Given a set of n (labeled) nodes, there are a total of $\binom{n}{2} = \dfrac{n(n-1)}{2}$ possible edges that can be placed between them. Hence, there are $2^{\binom{n}{2}}$ possible graphs on n nodes, since each edge can be either "on" or "off". To get an idea of how quickly this expression grows, note that it is approximately equal to 1000 for $n = 5$, but on the order of 3×10^{13} for $n = 10$. While the problem currently being dealt with does not require an exhaustive search of every possible graph on a given number of nodes, this example shows how quickly the number of possible graphs grows with the number of nodes and that explicit enumeration algorithms are doomed to fail because of the vastness of the search space.

2.2 Qualities of an Optimal Solution

When attempting to create an optimal configuration, there are many criteria that one may consider. For this project, we focused on the following.

- *connectivity* - maximizing the number of satellites that are connected to the main ground station.
- *components* - minimizing the number of connected components of the network is broken into, where a satellite/station cannot communicate with satellites/stations in another components,
- *maximum latency* - minimizing the maximum amount of time it takes for any one satellite to transmit data to the main ground station,
- *average latency* - minimizing the average amount of time it takes for each satellite to transmit data to the main ground station,
- *number of cut edges* - minimizing the number of edges that, when removed, increase the number of components of the graph,
- *number of cut nodes* - minimizing the number of nodes that, when removed, increase the number of components of the graph.

The relative importance of each of these criteria may vary, so a well designed algorithm would be able to return solutions that weight these criteria in different ways.

Chapter 3

Methods

In general, the problem of finding an optimal connection graph is difficult. Indeed, when $k = 2$, this is equivalent to the *Hamiltonian Path* problem (or *Hamiltonian Cycle* problem if we require redundancy), which is known to be NP-Complete [8]. Essentially, this means that any algorithm that finds an optimal configuration, or determines that there are no optimal configurations, must have a running time that is exponential in the number of vertices. When $k > 2$, it is unknown if the problem is NP-Complete, but even if it were not, additional difficulties would arise. With less stringent restrictions, it becomes easier to find a legal configuration, but it may be difficult to find a good one.

Let N_G be the total number of nodes in the graph. For each time step t_n our algorithm takes input in the form of a list of all possible pairwise satellite connections at t_n, along with their respective distances, d_{ij}. A weighted N_G x N_G *visibility matrix* is constructed by making the (i, j)th entry equal to d_{ij} if satellites i and j have a line of sight, and 0 otherwise. Note that this connection data changes with every minute, given the non-stationary trajectories of the satellites.

Our algorithm must process this data in two phases. First, it must find a number of good satellite connection configurations at each time step. For each time step, our algorithm returns these possible configurations as N_G x N_G *adjacency matrices* where the (i, j)th entry is equal to d_{ij} if satellites i and j should form an actual connection, and 0 otherwise. Our algorithm must then look through these possible configurations at each time step and find the best way to transition between them, taking into account non-trivial transition times.

Configurations are ranked on connectivity, latency, and redundancy, as described in Section 3.1, a ranking which is used by all further processes. Section 3.2 describes how random acceptable network configurations are found at each time step. Section 3.3 introduces the Genetic Search Algorithm that finds better configurations than a pure random search. Using the configurations found with the Genetic Search, Section 3.4 discusses how the algorithm chooses which configuration to use at each time, while taking transitions between configurations into account.

3.1 Scoring Configurations

In order to compare the quality of configurations, we establish a metric by which to score and rank configurations that takes into account the criteria in Section 2.2. We label the criteria S_i for $(i = 1 \ldots 6)$ as follows:

- S_1 = number of satellites disconnected from the main ground station,

- S_2 = number of components of the network,
- S_3 = maximum latency from any satellite (in milliseconds),
- S_4 = average latency from satellites to main ground station (in milliseconds),
- S_5 = number of edges that, when removed, break the graph into more components,
- S_6 = number of satellites that, when removed, break the graph into more components.

We then penalize each of these measurements by multiplying each S_i (i=1 ... 6) by a certain value. Though these numbers can be changed, here are our default values:

$$\text{score } = 100000S_1 + 1000S_2 + 20S_3 + 10S_4 + 2S_5 + 1S_6 \tag{3.1}$$

Equation 3.1 shows how we prioritize different configurations. The best configurations (with the lowest score) will have high connectivity and few components. Low latency is the next most important trait, and high redundancy is used to break ties.

3.2 Random Branching Sub-algorithm

As shown above, it is computationally infeasible to find every subgraph of a network that conforms to a set of degree constraints. Thus, in order to find the optimal or close-to-optimal connection configuration for any network, it is necessary to examine only a sampling of all acceptable subgraphs.

To create a sampling of all possible subgraphs, we use a branching approach that starts with an edgeless graph and fills it with randomly selected edges. These randomly selected edges conform to the degree constraints and the line of sight topology given by the visibility matrix. Our approach continues to connect potential neighbors until no more such connections can be made. Hence, it never returns a configuration that is a subgraph of another configuration. Because adding connections will only improve a network, this guarantees that we never return a configuration that is obviously an inferior subgraph of another configuration. This sub-algorithm can also build from a "framework" of specified edges, a fact that will be used later.

3.3 Genetic Search

A purely random search is not an efficient approach because it would take too long to sample an appreciable fraction of the large number of possible network configurations. We decided to take a more sophisticated approach to the problem, one that uses a random search as a subroutine, and that finds an optimal configuration faster. We called this algorithm "Genetic Search".

3.3.1 How Genetic Search Works

Genetic Search takes in a visibility graph and the cross link constraints, and returns a list of configurations for that time subject to the constraints. It also takes in 5 parameters, c_i for $(i = 1 \dots 5)$ which govern the running time and quality of configurations. They govern the following aspects of our algorithm:

- c_1 = number of attempts at generating new configurations in each iteration,

- c_2 = number of configurations kept per time,
- c_3 = number of configurations used to create the intersection,
- c_4 = number of best solutions that are checked for improvement,
- c_5 = number of iterations in a row without improvement that the algorithm will run before ending.

The role of these parameters will become clear in the explanation below.

Creating the Initial List

Genetic Search fist creates an initial list of configurations. There are three types of configurations in this list: configurations based on shortest paths, configurations from the previous time, and random configurations.

For each node in the network, our algorithm determines the shortest path between it and the ground station. The Random Branching sub-algorithm builds from this path, so edges are added in a random fashion until no more can be added without violating the cross link constraints. Since we do this for each node, it ensures that in our initial list of configurations, there is at least one configuration that contains every shortest path. This helps to ensure that we are not leaving out a path (which would minimize the maximum latency). It will create roughly $c_1/2$ configurations in this way.

Next, the algorithm looks at the c_2 best solutions from the previous time and updates them for the conditions of this time. It changes the distances to reflect the altered positions of the satellites, and if any connections that were used are no longer viable it removes them. While in some cases changing conditions will cause these configurations to be of low quality, most of the time (especially in high orbits) these configurations will be of comparable quality to the previous times.

Finally, the Random Branching Algorithm is run without a template $c_1/2$ times, creating a set of random configurations. The total number of configurations generated during this intimal setup is roughly c_1.

The Iterative Step

When given a list of configurations, the iterative step consists of the following. The algorithm starts by scoring the input list according to the discussion in Section 3.1, and finding the best c_2 configurations (see Figure 3.1). It then finds the intersection of the top c_3 of these c_2 configurations. That is, it finds the edges that are present in all of the best c_3 configurations (see Figure 3.2). These preserved edges are kept as a template upon which additional edges can be randomly added to find more network configurations.

The genetic search then outputs a new list consisting of three types of configurations: the original c_2 best configurations from the input list, a set of new configurations built upon the preserved edge template, and a set of random configurations $c_1/2$ times from the intersection, and an additional $c_1/2$ times with no initial edges preserved. These last configurations built without the intersection are included to ensure that keeping the edge intersection is not excluding other possible optimal configurations that may not have these preferred edges.

This new list of configurations is then used as the input list at the start the of next iteration. By always keeping the best configurations from the last iteration, the quality of configurations will improve or stay constant with each iteration. After each iteration, the

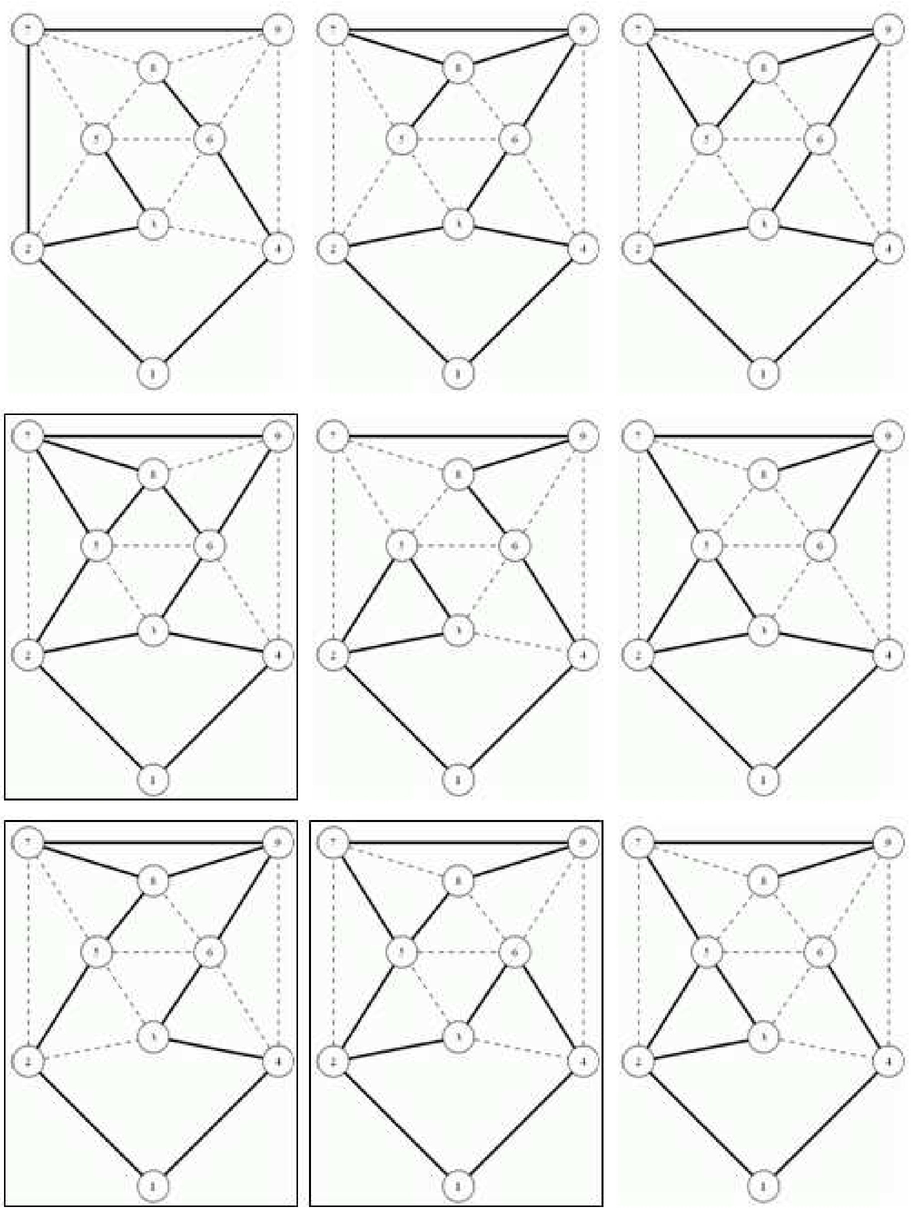


Figure 3.1: First part of the iterative step. A list of configurations is taken as input and the best configurations (those in boxes) are identified.

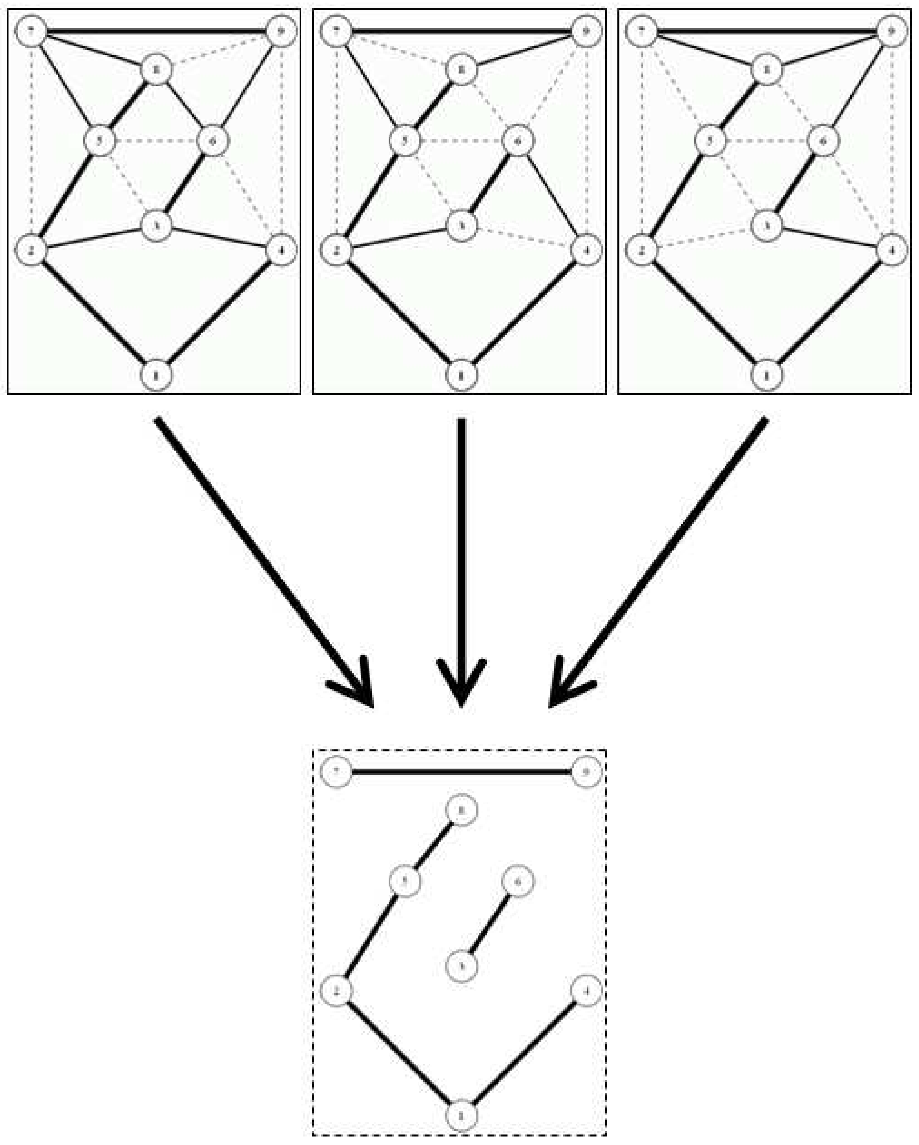


Figure 3.2: Second part of the iterative step. The intersection of the best few configuration is created.

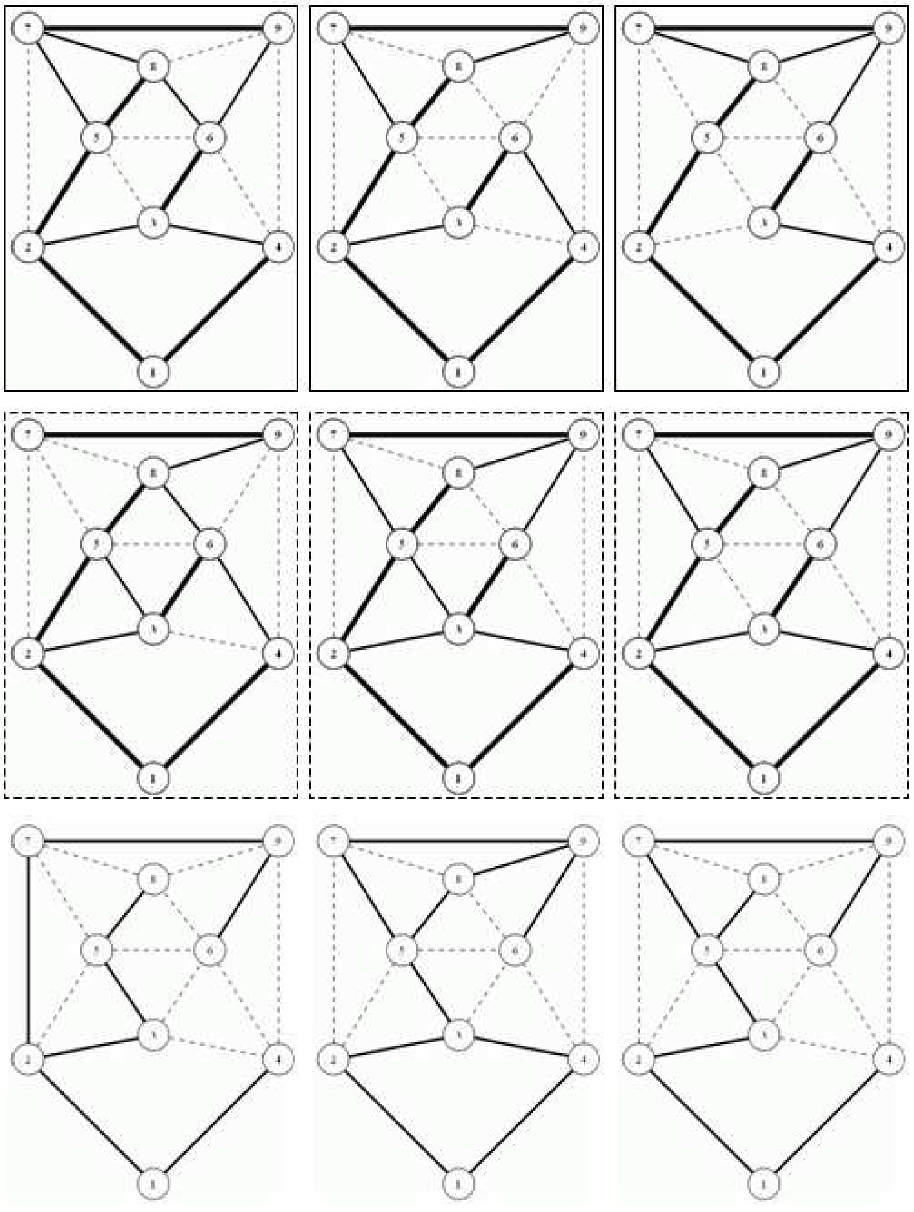


Figure 3.3: Third part of the iterative step. A new list of configurations is formed, consisting of the best configurations from the input (shown in solid boxes), configurations built from the intersection (shown in dashed boxes), and randomly generated configurations.

algorithm checks if there is any improvement to the best c_4 configurations. It will continue to iterate the above process, until it runs c_5 times in a row with no improvement to the best c_4 configurations.

3.3.2 Why Genetic Search Works

We believe that building upon the edges formed from the intersection of the best solutions leads to better solutions in the next iteration. There are two intuitive explanations for this. One deals with connectivity and one deals with latency.

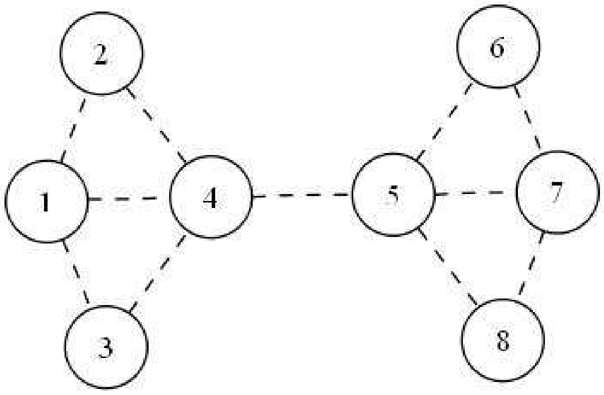
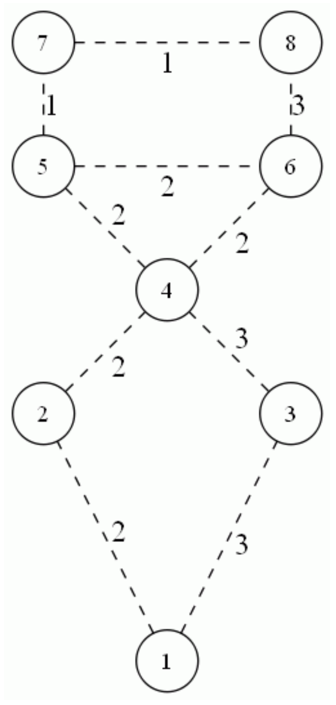


Figure 3.4: Simple portion of a network showing eight nodes. Each node may have 3 connections coming from it. Nodes 4 and 5 thus cannot connect to all of their neighbors.

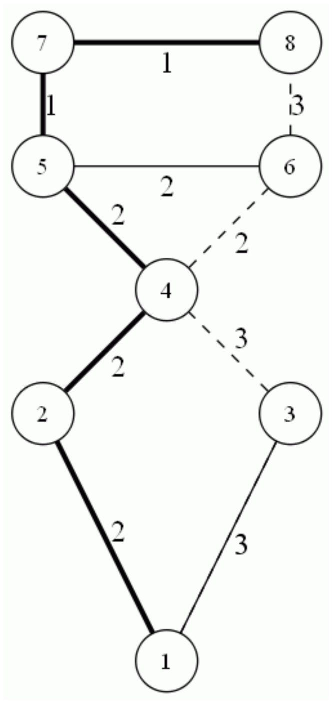
Since all the best solutions are going to be maximally connected, they must contain any critical edges in the satellite network. For example, consider the potential connections shown for the small network in Figure 3.4. This small network of eight nodes could be a portion of a larger, more complex network. If this network had a maximum degree of 3, both nodes 4 and 5 would need to forgo one of their possible connections to create an allowable subgraph. However, the edge between nodes 4 and 5 is critical to keeping the graph connected. All maximally connected solutions will keep the edge from node 4 to node 5. Therefore the intersection of the best solutions will contain this edge and ensure maximum connectivity.

Among equally connected solutions, low latency becomes the most important measure of a good network. The intersection of all good network designs will preserve the edges that lead to the shortest paths occurring between the ground station and outlying satellites.

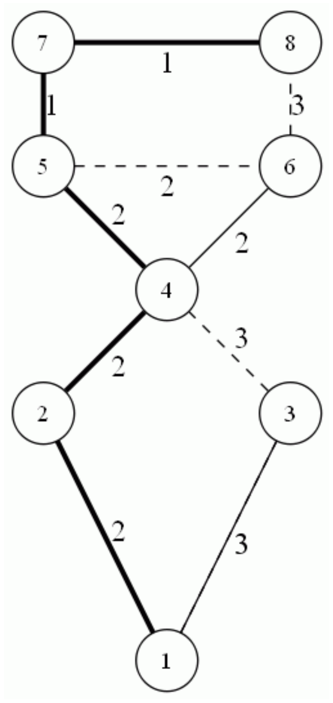
Consider the line of sight graph shown in Figure 3.5(a). Note that in this graph, node 8 is the farthest from the ground station, and there is a unique shortest path connecting it to the ground station. Therefore, any good configuration that minimizes the maximum latency will contain this shortest path, such as those shown in Figures 3.5(b) and 3.5(c). The intersection of (b) and (c) contains this shortest path, as would any new configuration built from the intersection. This principle generalizes not just to preserving the shortest path to the farthest satellite, but to others as well. By keeping edges critical to preserving high connectivity and low latency, the genetic search narrows in on the best edges that should be connected in the optimal solution to a network.



(a) The Line of Sight Graph



(b) A Possible Configuration



(c) Another Possible Configuration

Figure 3.5: (a) A line of sight graph. In this example, node 8 is the farthest from the ground station, and there is a unique shortest path connecting it to the ground station. Figures (b) and (c) represent possible configurations containing this shortest path, shown in bold. Therefore, the intersection of (b) and (c) would contain the shortest path to node 8, as would any new configurations built from the intersection.

3.4 Determining the Best Configurations Over Time

The search algorithms discussed so far are designed to find an optimal configuration for a fixed time, with unchanging potential connections between satellites. However, due to the dynamic motion of the satellites through time, the line-of-sight among the satellites is always changing. Hence, optimal configurations need to be found at each time. The simplest approach is to discretize time by minutes, so a day of satellite motion consists of 1440 consecutive graphs. At first glance, it would appear to be sufficient to simply find an optimal configuration at each time step. What this approach fails to take into account, however, is the transition time between two subsequent network configurations.

During a transition from one configuration to another, certain cross links are removed and others are added. To illustrate this process, consider Figure 3.6, where a configuration for one minute is transitioning to a configuration for the next minute. While transitioning, the original cross links are off, and the new ones have not yet come online. This switching means that throughout the duration of the transition that link is not active at all. Only those cross link edges which remain constant between subsequent configurations are present in the transition configuration. Therefore, while the configurations as shown in Figure 3.6 are both well-connected, the same cannot be said for the transition configuration. If we want our network to remain connected at all times and communication to satellites to be uninterrupted, this type of disconnection is unacceptable.

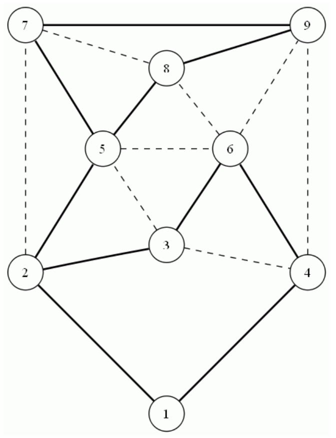
We start our over-time approach by finding a list of configurations for each time step. These lists of configurations, for each time step, are found independently with the Genetic Search Algorithm discussed earlier. If we visualize each of these configurations as a single node and align configurations from the same time in columns, we arrive at Figure 3.7. In this graph, there are three best configurations at time one, labeled A1, A2, and A3. Each of these configurations can transition to one of the three best configurations at time two, labeled B1, B2, and B3. Similarly, the configurations at time two can transition to the configurations at time three.

To find the best way to transition through Figure 3.7, scores are assigned to each path from one configuration to the next. These scores take into account both the starting configuration, such as Figure 3.6(a), and the transition configuration as the between solutions, such as Figure 3.6(c). Scores are assigned to each of these configurations according to Section 3.1 and then normalized for how long the configuration will be present. For example, if it takes 20 seconds to transition between the configurations represented in Figure 3.7, the total score of transitioning, say, from A1 to B2 would be

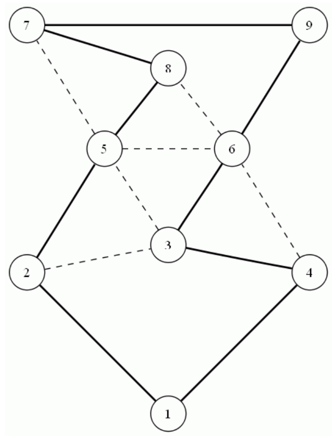
$$\text{score of transition} = \text{score(A1)}\frac{40}{60} + \text{score(A1} \cap \text{B2)}\frac{20}{60}.$$

The score of A1 is normalized by 40/60 because it is calculated assuming a full minute of time in a configuration, but A1 is only constant for 40 seconds before starting to transition to B2. The intersection of A1 and B2 is present during the 20 second transition, so the score of this configuration is multiplied by 20/60.

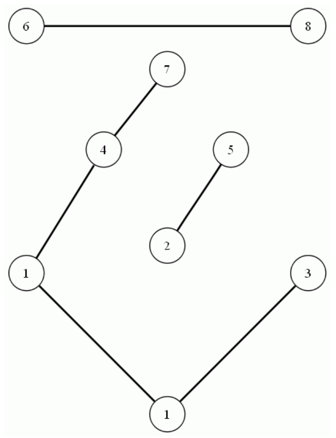
By repeating this calculation for every possible transition between two subsequent configurations, it is possible to add edge weights to all of the edges in Figure 3.7. The best way to transition through the possible configurations is simply the shortest path through all of these configurations and transitions. Finding this shortest path is simply an application of Dijikstra's Shortest Path Algorithm [3].



(a) Configuration at Time 1.



(b) Configuration at Time 2.



(c) Configuration during transition.

Figure 3.6: (a) A possible configuration at Time 1, where connections are shown as a solid line and unconnected satellites in range of each other are shown as a dotted line. (b) Possible configuration at Time 2. (c) Configuration during the transition from the configuration at Time 1 to the configuration at Time 2. Note how only those edges which remain constant are present in the transition. While the configurations at Time 1 and Time 2 are connected, the transition configuration has disconnected satellites. Communication to these satellites will be disrupted during the transition.

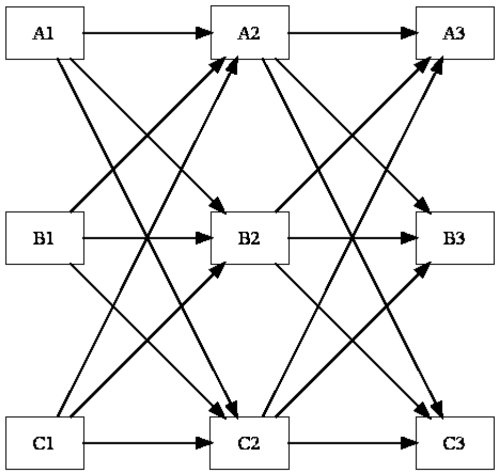


Figure 3.7: Three possible configurations (A1, B1, and C1) are given for time 1. Each of these can transition to one of three configurations (A2, B2, or C2) at time two, which in turn can transition to (A3, B3, or C3) at time three.

3.4.1 Transition Times of More than One Minute

Figure 3.7 represents how our algorithm works for transition times of less than one minute, where it is easy to be at one configuration at time one and a different configuration at time two. However, if it takes more than one minute to transition between configurations, going from a configuration at one time step to a different configuration at the next is impossible. In this case, looking ahead more than one time step is necessary. For example, if we start at a configuration for time one and the transition time is two and half minutes, the next available configurations would be those at time four. We would stay in the first configuration for 30 seconds, then spend two and a half minutes transitioning.

The caveat with transition times of more than one minute is that the transition configuration must remain feasible during the intervening minutes. If transitioning from configuration A1 at time one to configuration D4 at time four, then the configuration A1 $\bigcap$ D4 must be a subgraph of the visibility graph at times two and three. Usually, this is not a problem because satellites in fixed orbits are unlikely to go in or out of range with such a frequency, but it is something to take into account. Transition configurations are typically worse with regard to connectivity, latency, and redundancy than the best configurations at a specific time. This is because a transition configuration contains only those edges preserved between the two subsequent configurations. The longer the transition time, the longer they will have to remain in these inferior transition configurations.

3.4.2 Configurations that Remain Feasible

Because transition configurations are inferior to the best configurations at a given time, it is preferable to minimize the number of transitions that must be made. Consider again Figure 3.7. If the cross links used in configuration A1 are present in the visibility matrix of time two, configuration A1 would be able to remain unchanged at time two, and then transition straight to a new configuration at time three. Our phase II algorithm takes this into account by updating the graph represented in Figure 3.7 to that shown in the Figure 3.8.

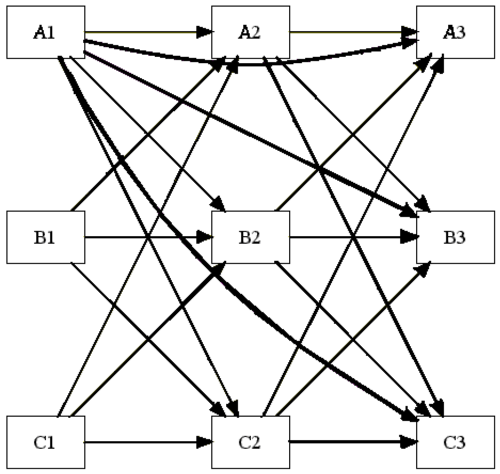


Figure 3.8: In Figure 3.7, satellite configurations had to transition to a new configuration at each time step. However, if a configuration remains viable at the next time, then we add edges in order to allow for fewer transitions. Above we show what edges would be added (shown in bold) if configuration A1 remained a viable option at time 2.

The score of that path (if the transition time were 20 seconds) is then calculated in the following way.

$$\text{score }(\text{A1} \rightarrow \text{C2}) = \text{score}_{t_1}(\text{A1})\frac{60}{60} + \text{score}_{t_2}(\text{A1})\frac{40}{60} + \text{score}_{t_2}(\text{A1} \cap \text{C2})\frac{20}{60}.$$

By $\text{score}_{t_2}(\text{A1})$ we mean the score of configuration A1, after being updated to have the correct lengths of the connections that exist at t_2. In this way, our edge weight calculation takes into account the fact the dynamic nature of the satellites, and accounts for the fact that a configuration at one time will be different.

By looking at each configuration and seeing how far into the future it remains viable, it is simple to add edges (as in Figure 3.8) to give configurations the options of remaining constant for more than a single time. Once all these edges are added, and scores for all the transitions have been calculated, then it is just a matter of finding the shortest path from configurations at time one to the configurations at the last time. To carry out this procedure, we used Dijikstra's shortest path algorithm which quickly finds the shortest path through this graph, corresponding to choosing the configurations at each time that minimizes the overall score.

Chapter 4

Results

In order to ensure the performance and adaptability of our algorithm, we ran several types of tests.

An initial test was to construct a satellite network consisting of six nodes fully connected, where $d_{ij} > 0$ for all $i, j \in 1, 2, 3, 4, 5, 6$. We ran a "brute force" enumeration of the 2^{15} possible subgraphs, and identified the best one. We then ran our genetic search algorithm, which also found the optimal configuration.

However, in general such an approach is an impractical method of verifying the quality of a configuration. It was only possible in this case because the size of the network was small enough that an enumeration approach was feasible. For larger networks, we have adapted another way of evaluating our results.

4.1 The Ideal Configuration

In cases where a brute force enumeration is not feasible, we compare the score (as defined in section 3.1) of our generated configuration to that of the "ideal configuration" at that time. The ideal configuration is constructed by taking the subgraph of the line of sight graph consisting of every edge that could potentially exist. While in most cases this will be the line of sight graph itself, it may be the case that while two satellites have a line of sight between them, they do not have the same type of cross link equipment, so they would never be able to communicate directly.

We then compute the latency, redundancy, and other qualities for this ideal configuration. These qualities, which we denote the "ideal qualities" for any solution, serve as a bound on achievable values. This is because for any configuration, the connectivity, redundancy, or shortest path between nodes is never improved by removing edges, and any configuration our algorithm returns will necessarily be a subgraph of the ideal configuration. It is important to recognize that these ideals are a bound, not a goal. In fact, for many networks with cross link constraints it is impossible for any solution to achieve these ideals.

For example, in some cases it may be impossible to achieve ideal connectivity. Consider the network shown in Figure 4.1. If the restrictions are such that each satellite is only allowed one downlink and two cross links, then we can see that there is no way for every satellite to be connected to the ground station (represented by node 1). The best that could be done for this network would be to connect the ground station and Satellite 2, and then any of the two others. In fact, in any case where there is a similar "star-like" configuration, where the node in the center has more potential links than allowed links, then there is no way for any configuration to achieve ideal connectivity. This is just an example of one

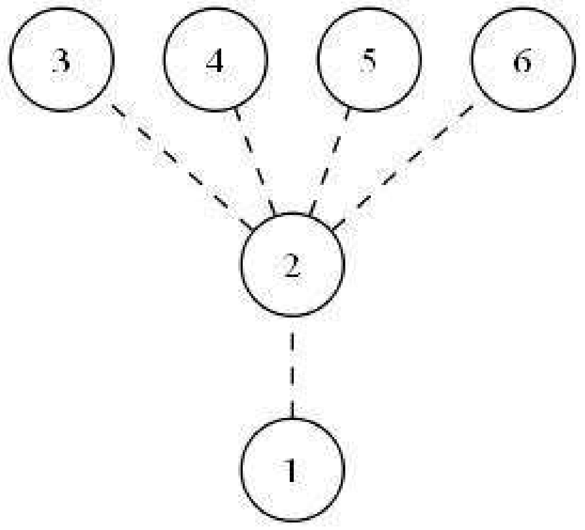


Figure 4.1: If the cross link restrictions are such that each satellite is allowed three cross links, and one down link, then we see that it is impossible for satellite two to make every possible connection. Therefore any configuration obeying the cross link restrictions will fall short of the ideal.

situation where ideal connectivity cannot be achieved; there are other configurations aside from these "star-like" situations.

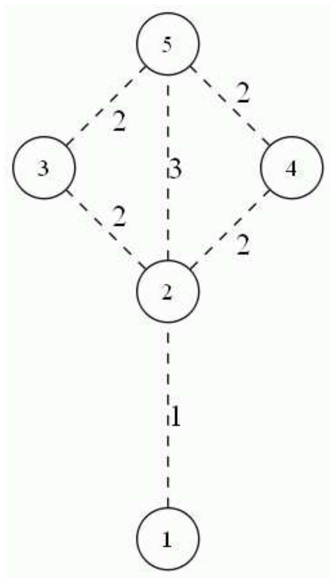
In other cases, it is impossible to achieve ideal latency. Consider the network in Figure 4.2(a). If we consider this network with the same cross link restrictions as the above example it becomes impossible to achieve the ideal maximum latency. In the ideal configuration, represented by Figure 4.2(b), the satellite farthest from the ground station is Satellite 5, which is a distance of 4 units away. However, in Figure 4.2(c), we see the best possible configuration, subject to the given constraints. Under these connections, Satellite 5 is a distance of 5 units away, which is greater than in the ideal case. This is just one small example of a case where the ideal is not achievable.

Comparing against the ideal is an imperfect metric, for it sets unattainable goals. However, in networks with somewhat relaxed cross link restraints, it is encouraging to see how close our algorithm's results come to the ideal, illustrating that it is in fact finding very high quality solutions.

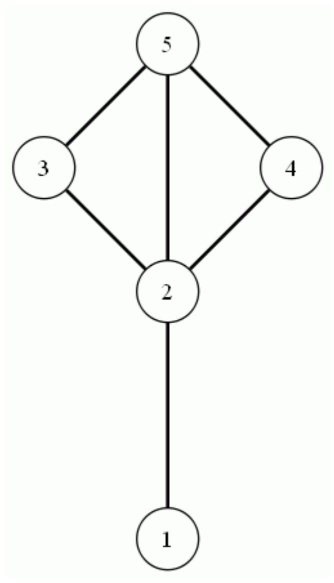
In general, configurations created subject to restrictive restraints are furthest from the ideal, whereas those with looser constraints will come closer.

4.2 Results Using Test Data

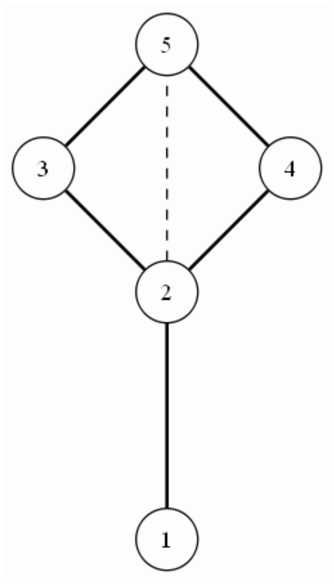
Data provided by the Aerospace Corporation consist of test satellite configurations. These included one network of satellites in Medium Earth Orbits (MEOs), and two networks of satellites in Low Earth Orbits (LEOs). While in these examples all satellites were of the same orbit type, our algorithm is designed to deal with the case of multiple types of orbits in the same network. For each network, Aerospace provided four different scenarios of the cross link equipment contained on each satellite. In the first scenario, every satellite was allowed two cross links to other satellites ($k_{max} = 2$), and one down link to the ground station. In the second scenario, every satellite was allowed three cross links to other satellites



(a) The Line of Sight Graph



(b) The Ideal Configuration



(c) An Optimal but Subideal Solution

Figure 4.2: An example of a configuration where it is impossible to achieve the ideal maximum latency. In the ideal configuration, (b), the satellite farthest from the ground station is Satellite 5, which is a distance of 4 units away. However, in (c), we see the best possible configuration, subject to the constraints. Under these connections, Satellite 5 is a distance of 5 units away, which is greater than in the ideal case.

($k_{max} = 3$), and one down link to the ground station. In the third scenario, every satellite was allowed three cross links to other satellites, and one down link to the ground station, but there were two distinct types of cross links. In the fourth scenario, every satellite was allowed three cross links to other satellites, and one down link to the ground station, but there were two distinct types of cross links and varying types of down links. Specific details on the various scenarios can be found in Appendix B.

In the following tables, several statistics from these tests are given. For each of the 3 constellations, we had the 4 cross link scenarios, resulting in a total of 12 distinct constellations to consider. For each of those, we considered four possible transition times: 0 seconds, 30 seconds, 1 minute, and 2 minutes. The 0 second transition time corresponds to taking the best configuration at each time, whereas the non-trivial transition times consider the state of the network over time.

For each results chart (Tables 4.1 - 4.12), there are three subsections of data: Connectivity, Maximum Latency, and Average Latency. In Connectivity, there are two values. We counted the number of configurations (out of the possible 1440, corresponding to one for each minute of data) which attain the ideal number of satellites connected to the ground station. We also count how many configurations have the ideal number of components. These quantities are related, but distinct. For example, in certain constellations, there are times where there are no connections possible between the ground stations and satellites. During these times, our algorithm still connects the satellites together, despite there being no connection to the ground station. Therefore, we would hope in a situation such as that, there would be only 2 components in our configurations, one corresponding to the ground stations, and another corresponding to the satellites.

For Maximum Latency, we display several statistics. First, we have the average difference (across the 1440 times) between the ideal and achieved maximum latency, in milliseconds. We also display that same aspect, but as a percentage difference from the ideal. In order to give an idea of how many configurations are achieving close to the ideal, we also give a count of how many configurations achieve the ideal, are within 1% of the ideal, are within 5% of the ideal, and are within 10% of the ideal. We also display all of this information for Average Latency. In scenarios where the maximum number of connected satellites is not always achieved, these statistics are computed only considering those configurations which do achieve this. That is because it would be meaningless to compare the average latency from the ground station to 5 satellites to an ideal configuration's average over 8 satellites, for example.

There are a few important things to take into account when viewing these results. The first, as has been mentioned before, is that one should not expect to achieve the ideal configuration, especially in scenarios with restrictive cross link restraints. For example, on each constellation the results for Scenario 1 are far below the ideal. This is because Scenario 1 only allowed two cross links per satellite, and so there it is impossible to include anywhere near all the cross links contained in the ideal configuration. Furthermore, it is worth noting that for a single scenario, as the transition time goes up, the quality of configurations used goes down. These statistics are provided not so much to be viewed on their own, but as a standard to be compared against for any other algorithm.

For all of these test runs, we used the same search parameters, as described in section 3.3.1. We provide the values we used for the test as well as a short reminder for each of them below:

- $c_1 = 100$ (number of attempts at generating new configurations in each iteration),
- $c_2 = 10$ (number of configurations kept per time),

- $c_3 = 3$ (number of configurations used to create the intersection),
- $c_4 = 3$ (number of best solutions that are checked for improvement),
- $c_5 = 5$ (number of iterations in a row without improvement that the algorithm will run before ending).

Small LEO - Scenario 1

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1440 (100%)	1440 (100%)	1433 (99%)	1423 (99%)
Configs at Min # of Comps	1440 (100%)	1440 (100%)	1433 (99%)	1423 (99%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	10.8	20.8	22.1	24.4
Average % Difference	10.5%	21.0%	22.2%	24.4%
Maximum Difference (ms)	53.9	82.4	214.1	262.0
Maximum % Difference	49.2%	81.8%	204.2%	224.0%
Configs at Ideal	454 (32%)	186 (13%)	188 (13%)	148 (10%)
Configs Within 1%	518 (36%)	211 (15%)	204 (14%)	170 (12%)
Configs Within 5%	826 (57%)	342 (24%)	340 (24%)	304 (21%)
Configs Within 10%	996 (69%)	477 (33%)	484 (34%)	457 (32%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	2.9	5.7	6.5	7.0
Average % Difference	4.6%	9.9%	11.3%	12.2%
Maximum Difference (ms)	22.7	82.4	78.6	93.2
Maximum % Difference	33.0%	81.7%	123.3%	127.0%
Configs at Ideal	282 (20%)	118 (8%)	109 (8%)	75 (5%)
Configs Within 1%	620 (43%)	212 (15%)	189 (13%)	158 (11%)
Configs Within 5%	989 (69%)	506 (35%)	481 (33%)	478 (33%)
Configs Within 10%	1287 (89%)	944 (66%)	908 (63%)	866 (60%)

Table 4.1: Results for Small LEO - Scenario 1

Small LEO - Scenario 2

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1440 (100%)	1440 (100%)	1432 (99%)	1432 (99%)
Configs at Min # of Comps	1440 (100%)	1440 (100%)	1432 (99%)	1432 (99%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	0.1	0.5	0.9	1.5
Average % Difference	0.2%	0.5%	1.0%	1.5%
Maximum Difference (ms)	14.6	15.5	67.5	59.8
Maximum % Difference	16.0%	16.0%	72.8%	60.5%
Configs at Ideal	1377 (96%)	1185 (82%)	1212 (84%)	1142 (79%)
Configs Within 1%	1386 (96%)	1242 (86%)	1265 (88%)	1199 (83%)
Configs Within 5%	1423 (99%)	1410 (98%)	1367 (95%)	1334 (93%)
Configs Within 10%	1437 (100%)	1435 (100%)	1402 (97%)	1376 (96%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	0.3	0.5	0.8	1.0
Average % Difference	0.4%	0.8%	1.4%	1.8%
Maximum Difference (ms)	4.5	17.5	32.0	27.5
Maximum % Difference	8.5%	29.9%	57.3%	54.2%
Configs at Ideal	1173 (81%)	881 (61%)	856 (59%)	767 (53%)
Configs Within 1%	1230 (85%)	1052 (73%)	1088 (76%)	1005 (70%)
Configs Within 5%	1427 (99%)	1411 (98%)	1370 (95%)	1337 (93%)
Configs Within 10%	1440 (100%)	1432 (99%)	1385 (96%)	1369 (95%)

Table 4.2: Results for Small LEO - Scenario 2

Small LEO - Scenario 3

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1440 (100%)	1440 (100%)	1438 (100%)	1438 (100%)
Configs at Min # of Comps	1440 (100%)	1440 (100%)	1438 (100%)	1438 (100%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	1.6	2.1	4.0	4.9
Average % Difference	1.5%	1.9%	3.6%	4.3%
Maximum Difference (ms)	32.3	36.8	103.1	79.4
Maximum % Difference	40.7%	40.7%	101.0%	73.0%
Configs at Ideal	1158 (80%)	985 (68%)	882 (61%)	802 (56%)
Configs Within 1%	1198 (83%)	1049 (73%)	941 (65%)	863 (60%)
Configs Within 5%	1318 (92%)	1288 (89%)	1192 (83%)	1136 (79%)
Configs Within 10%	1362 (95%)	1350 (94%)	1284 (89%)	1246 (87%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	0.7	1.0	1.8	2.1
Average % Difference	1.3%	1.8%	2.9%	3.5%
Maximum Difference (ms)	12.2	13.1	55.2	38.9
Maximum % Difference	20.6%	20.9%	110.5%	75.7%
Configs at Ideal	756 (53%)	569 (40%)	470 (33%)	395 (27%)
Configs Within 1%	1023 (71%)	827 (57%)	728 (51%)	644 (45%)
Configs Within 5%	1331 (92%)	1286 (89%)	1212 (84%)	1157 (80%)
Configs Within 10%	1411 (98%)	1406 (98%)	1350 (94%)	1332 (93%)

Table 4.3: Results for Small LEO - Scenario 3

Small LEO - Scenario 4

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1440 (100%)	1440 (100%)	1438 (100%)	1436 (100%)
Configs at Min # of Comps	1440 (100%)	1440 (100%)	1438 (100%)	1436 (100%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	2.3	3.6	5.2	6.2
Average % Difference	2.1%	3.3%	4.7%	5.5%
Maximum Difference (ms)	32.3	48.3	103.1	79.4
Maximum % Difference	40.7%	47.9%	101.0%	73.0%
Configs at Ideal	995 (69%)	755 (52%)	690 (48%)	646 (45%)
Configs Within 1%	1065 (74%)	831 (58%)	759 (53%)	711 (49%)
Configs Within 5%	1259 (87%)	1147 (80%)	1075 (75%)	1016 (71%)
Configs Within 10%	1338 (93%)	1285 (89%)	1223 (85%)	1185 (82%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	2.4	2.7	3.5	3.8
Average % Difference	4.2%	4.9%	6.1%	6.7%
Maximum Difference (ms)	16.2	15.0	55.2	38.9
Maximum % Difference	27.2%	29.9%	110.5%	75.7%
Configs at Ideal	451 (31%)	338 (23%)	282 (20%)	236 (16%)
Configs Within 1%	578 (40%)	465 (32%)	419 (29%)	354 (25%)
Configs Within 5%	985 (68%)	929 (65%)	878 (61%)	822 (57%)
Configs Within 10%	1251 (87%)	1227 (85%)	1143 (79%)	1114 (77%)

Table 4.4: Results for Small LEO - Scenario 4

Large LEO - Scenario 1

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	508 (35%)	456 (32%)	331 (23%)	297 (21%)
Configs at Min # of Comps	313 (22%)	261 (18%)	153 (11%)	134 (9%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	11.6	15.0	14.0	13.1
Average % Difference	15%	19.6%	18.9%	16.3%
Maximum Difference (ms)	100.2	100.2	106.6	84.6
Maximum % Difference	165.9%	165.9%	210.8%	155.1%
Configs at Ideal	216 (15%)	212 (15%)	186 (13%)	170 (12%)
Configs Within 1%	224 (16%)	217 (15%)	190 (13%)	175 (12%)
Configs Within 5%	236 (16%)	233 (16%)	206 (14%)	186 (13%)
Configs Within 10%	308 (21%)	272 (19%)	222 (15%)	202 (14%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	2.8	3.7	3.4	3.4
Average % Difference	7.4%	10.0%	9.2%	8.6%
Maximum Difference (ms)	27.4	28.8	29.2	22.7
Maximum % Difference	92.7%	97.6%	104.0%	79.3%
Configs at Ideal	210 (15%)	208 (14%)	183 (13%)	167 (12%)
Configs Within 1%	227 (16%)	219 (15%)	190 (13%)	176 (12%)
Configs Within 5%	263 (18%)	251 (17%)	210 (15%)	188 (13%)
Configs Within 10%	383 (27%)	312 (22%)	232 (16%)	214 (15%)

Table 4.5: Results for Large LEO - Scenario 1

Large LEO - Scenario 2

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1440 (100%)	1440 (100%)	1207 (84%)	1112 (77%)
Configs at Min # of Comps	1440 (100%)	1440 (100%)	1164 (81%)	1052 (73%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	.01	.29	2.2	4.0
Average % Difference	.02%	.32%	2.6%	4.5%
Maximum Difference (ms)	2.7	67.2	85.9	96.2
Maximum % Difference	5.5%	82.3%	142.2%	123.6%
Configs at Ideal	1408 (98%)	1301 (90%)	940 (65%)	797 (55%)
Configs Within 1%	1435 (99.7%)	1370 (95.1%)	997 (69.2%)	863 (59.9%)
Configs Within 5%	1439 (99.9%)	1413 (98.13%)	1092 (75.8%)	958 (66.5%)
Configs Within 10%	1440 (100%)	1422 (98.8%)	1133 (78.7%)	989 (68.7%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	.12	.25	1.2	2.1
Average % Difference	.28%	.57%	2.7%	4.5%
Maximum Difference (ms)	4.4	24.1	38.2	52.7
Maximum % Difference	11.6%	60.8%	119.8%	139.1%
Configs at Ideal	702 (49%)	657 (46%)	459 (32%)	379 (26%)
Configs Within 1%	1385 (96.2%)	1346 (93.47%)	972 (67.5%)	804 (55.8%)
Configs Within 5%	1425 (99.0%)	1399 (97.1%)	1043 (72.4%)	883 (61.3%)
Configs Within 10%	1437 (99.8%)	1417 (98.4%)	1094 (76.0%)	972 (67.5%)

Table 4.6: Results for Large LEO - Scenario 2

Large LEO - Scenario 3

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1230 (85%)	1223 (85%)	981 (68%)	938 (65%)
Configs at Min # of Comps	1129 (78%)	1122 (78%)	742 (52%)	706 (49%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	0.6	1.1	2.3	3.3
Average % Difference	0.8%	1.6%	3.4%	4.4%
Maximum Difference (ms)	50.2	61.3	61.3	65.3
Maximum % Difference	100.6%	100.6%	100.6%	106.9%
Configs at Ideal	1177 (82%)	1045 (73%)	762 (53%)	688 (48%)
Configs Within 1%	1199 (83%)	1102 (77%)	801 (56%)	736 (51%)
Configs Within 5%	1208 (84%)	1165 (81%)	872 (61%)	807 (56%)
Configs Within 10%	1209 (84%)	1176 (82%)	900 (63%)	834 (58%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	0.3	0.6	1.3	1.7
Average % Difference	0.8%	1.9%	3.9%	4.9%
Maximum Difference (ms)	16.9	24.6	27.9	30.6
Maximum % Difference	42.5%	61.7%	78.3%	133.3%
Configs at Ideal	788 (55%)	680 (47%)	478 (33%)	440 (31%)
Configs Within 1%	1147 (80%)	1013 (70%)	699 (49%)	626 (43%)
Configs Within 5%	1184 (82%)	1112 (77%)	801 (56%)	737 (51%)
Configs Within 10%	1203 (84%)	1151 (80%)	856 (59%)	800 (56%)

Table 4.7: Results for Large LEO - Scenario 3

Large LEO - Scenario 4

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1221 (85%)	1215 (84%)	970 (67%)	941 (63%)
Configs at Min # of Comps	1120 (78%)	1113 (77%)	737 (51%)	723 (50%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	0.7	1.4	2.7	4.1
Average % Difference	1%	2.2%	3.8%	6.1%
Maximum Difference (ms)	50.2	50.2	50.4	70.8
Maximum % Difference	119%	119%	119%	150.6%
Configs at Ideal	1155 (80%)	976 (68%)	688 (48%)	629 (44%)
Configs Within 1%	1177 (82%)	1028 (71%)	718 (50%)	668 (46%)
Configs Within 5%	1191 (83%)	1108 (77%)	813 (57%)	752 (52%)
Configs Within 10%	1194 (83%)	1140 (79%)	862 (60%)	791 (55%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	0.8	1.1	1.9	2.3
Average % Difference	2.3%	3.6%	5.5%	7.2%
Maximum Difference (ms)	17.2	17.2	27.9	31.8
Maximum % Difference	81.5%	81.5%	81.5%	133.33%
Configs at Ideal	638 (44%)	562 (39%)	405 (28%)	372 (26%)
Configs Within 1%	924 (64%)	818 (57%)	572 (40%)	518 (36%)
Configs Within 5%	1062 (74%)	985 (68%)	691 (48%)	641 (46%)
Configs Within 10%	1123 (78%)	1063 (74%)	780 (54%)	726 (50%)

Table 4.8: Results for Large LEO - Scenario 4

MEO - Scenario 1

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1440 (100%)	1440 (100%)	1438 (100%)	1434 (100%)
Configs at Min # of Comps	1440 (100%)	1440 (100%)	1438 (100%)	1434 (100%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	6.8	37.1	33.5	36.2
Average % Difference	4.6%	24.6%	22.2%	24%
Maximum Difference (ms)	39.9	120.6	230.4	207.6
Maximum % Difference	26.3%	77.0%	156.5%	140.8%
Configs at Ideal	354 (24.6%)	9 (0.6%)	9 (0.6%)	9 (1%)
Configs Within 1%	532 (36.9%)	10 (0.7%)	10 (0.6%)	10 (1%)
Configs Within 5%	969 (67.3%)	94 (6.5%)	138 (9.6%)	106 (7%)
Configs Within 10%	1213 (84.2%)	218 (15.1%)	360 (25%)	245 (17%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	9.4	15.84	15.4	16.3
Average % Difference	11.3%	19.0%	18.5%	19.6%
Maximum Difference (ms)	20.6	45.8	93.7	74.3
Maximum % Difference	26.2%	57.5%	110.7%	86.1%
Configs at Ideal	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Configs Within 1%	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Configs Within 5%	111 (7.7%)	10 (0.7%)	28 (1.9%)	10 (1%)
Configs Within 10%	642 (44.6%)	240 (16.7%)	346 (24%)	263 (18%)

Table 4.9: Results for MEO - Scenario 1

MEO - Scenario 2

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1440 (100%)	1440 (100%)	1440 (100%)	1440 (100%)
Configs at Min # of Comps	1440 (100%)	1440 (100%)	1440 (100%)	1440 (100%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	0.9	3.7	3.7	4.4
Average % Difference	0.7%	2.5%	2.5%	3.0%
Maximum Difference (ms)	19.0	62.8	155.0	155.0
Maximum % Difference	13.9%	45.6%	104.3%	104.3%
Configs at Ideal	995 (69%)	654 (45%)	640 (44%)	557 (39%)
Configs Within 1%	1200 (83%)	865 (60%)	888 (62%)	774 (54%)
Configs Within 5%	1403 (97%)	1214 (84%)	1246 (87%)	1215 (84%)
Configs Within 10%	1438 (99%)	1326 (92%)	1341 (93%)	1338 (92%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	4.8	6.3	6.4	7.0
Average % Difference	5.7%	7.6%	7.7%	8.5%
Maximum Difference (ms)	15.9	27.6	46.9	46.9
Maximum % Difference	20.1%	35.0%	54.1%	54.1%
Configs at Ideal	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Configs Within 1%	28 (2%)	10 (1%)	4 (0%)	6 (0%)
Configs Within 5%	648 (45%)	466 (32%)	418 (29%)	413 (29%)
Configs Within 10%	1306 (91%)	1059 (74%)	1081 (75%)	1010 (70%)

Table 4.10: Results for MEO - Scenario 2

MEO - Scenario 3

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1440 (100%)	1440 (100%)	1440 (100%)	1440 (100%)
Configs at Min # of Comps	1440 (100%)	1440 (100%)	1440 (100%)	1440 (100%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	3.7	4.9	5.6	6.7
Average % Difference	2.4%	3.2%	3.6%	4.3%
Maximum Difference (ms)	36.1	41.4	179.4	180
Maximum % Difference	23.3%	29.3%	113.9%	114%
Configs at Ideal	855 (59.4%)	629 (43.7%)	680 (47.2%)	630 (43.8%)
Configs Within 1%	975 (67.7%)	789 (54.8%)	820 (56.9%)	760 (52.8%)
Configs Within 5%	1152 (80.0%)	1091 (75.7%)	1060 (73.6%)	1039 (72.2%)
Configs Within 10%	1317 (91.5%)	1267 (88%)	1242 (86.3%)	1212 (84.2%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	5.8	7	7.1	7.7
Average % Difference	6.9%	8.3%	8.4%	9%
Maximum Difference (ms)	16.5	18.1	40.9	48.7
Maximum % Difference	20.6%	23%	46.7%	55.5%
Configs at Ideal	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Configs Within 1%	55 (3.8%)	38 (2.6%)	0 (0%)	22 (1.5%)
Configs Within 5%	514 (35.7%)	388 (26.9%)	398 (27.6%)	333 (23.1%)
Configs Within 10%	1171 (81.3%)	1013 (70.4%)	1001 (69.5%)	984 (68.3%)

Table 4.11: Results for MEO - Scenario 3

MEO - Scenario 4

Connectivity				
	0 sec	30 sec	1 min	2 min
Configs at Max Connectivity	1440 (100%)	1440 (100%)	1440 (100%)	1440 (100%)
Configs at Min # of Comps	1440 (100%)	1440 (100%)	1440 (100%)	1440 (100%)

Maximum Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	5.1	8.3	9.1	10.7
Average % Difference	3.3%	5.3%	5.8%	6.8%
Maximum Difference (ms)	44.8	53.6	130.5	195.8
Maximum % Difference	31.8%	33%	84%	136.5%
Configs at Ideal	773 (53.7%)	507 (35.2%)	529 (36.7%)	480 (33.3%)
Configs Within 1%	877 (60.9%)	606 (42.1%)	618 (42.9%)	566 (39.3%)
Configs Within 5%	1090 (75.7%)	980 (68.1%)	938 (65.1%)	903 (62.7%)
Configs Within 10%	1243 (86.3%)	1172 (81.4%)	1153 (80.1%)	1118 (77.6%)

Average Latency				
	0 sec	30 sec	1 min	2 min
Average Difference (ms)	9.7	11.3	11.4	12
Average % Difference	11.3%	13.1%	13.3%	14%
Maximum Difference (ms)	35.7	39.4	54.2	55.6
Maximum % Difference	6%	23%	63.3%	64%
Configs at Ideal	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Configs Within 1%	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Configs Within 5%	212 (14.7%)	173 (12%)	156 (10.8%)	160 (11.1%)
Configs Within 10%	771 (53.5%)	558 (38.8%)	639 (44.4%)	574 (39.9%)

Table 4.12: Results for MEO - Scenario 4

Appendix A

Notes on the Parellizability of our Algorithm

When an algorithm is said to be parellelizable, this means many of the computations can be done independently of each other. That is, the computations can be split among many different processors and synthesized at the end. One advantage to our code is that the most computationally intensive aspects of it are parellelizable. Thus, it can run in a very short period of time if the work is split over many processors.

For example, a bulk of the running time of our algorithm is due to generating a list of good configurations via the Genetic Search algorithm. Since information about optimal configurations from one time step is used in subsequent time steps, each time step must be done one at a time. However, within a given time step, the time it takes to find a set of configurations can be greatly reduced via parallel computing. At each iteration, Genetic Search runs the Random Branching sub-algorithm c_3 times. Each of these sub-algorithms can be run independently. Assuming the default number of configurations generated at each iteration (100), the parallelized running time for this section could be reduced to $\frac{1}{100}$ of the normal running time.

The next step of the algorithm is going through the best configurations at each time step and computing the different scores of transitioning to each of the configurations at the next time step (or possibly further time steps if the graph is stable enough). In this case, not only can each configuration in a given time step be done independently, but each time step can be done independently. Assuming the default number of configurations returned for each time step (10), and 1440 time steps, the parallelized running time for this section could be reduced to $\frac{1}{14400}$ of the normal running time.

Appendix B

Scenario Details

This appendix contains the details about the link restrains applied in the various scenarios. The charts below display the maximum number each distinct type of link allowed for each satellite or ground station.

Table B.1: Connection Data for Small LEO - Scenario 1

Node	Cross Links	Up/Down Links
Satellite 111	2	1
Satellite 112	2	1
Satellite 113	2	1
Satellite 121	2	1
Satellite 122	2	1
Satellite 123	2	1
Satellite 131	2	1
Satellite 132	2	1
Satellite 133	2	1
Denver	0	2
Miami	0	2

Table B.2: Connection Data for Small LEO - Scenario 2.

Node	Cross Links	Up/Down Links
Satellite 111	3	1
Satellite 112	3	1
Satellite 113	3	1
Satellite 121	3	1
Satellite 122	3	1
Satellite 123	3	1
Satellite 131	3	1
Satellite 132	3	1
Satellite 133	3	1
Denver	0	2
Miami	0	2

Table B.3: Connection Data for Small LEO - Scenario 3.

Node	Cross Link-A	Cross Link-B	Up/Down Links
Satellite111	3	0	1
Satellite112	2	1	1
Satellite113	0	3	1
Satellite121	3	0	1
Satellite122	1	2	1
Satellite123	0	3	1
Satellite131	3	0	1
Satellite132	2	1	1
Satellite133	0	3	1
Denver	0	0	2
Miami	0	0	2

Table B.4: Connection Data for Small LEO - Scenario 4.

Node	Cross Link-A	Cross Link-B	Down Link-A	Down Link-B
Satellite111	3	0	1	0
Satellite112	2	1	1	0
Satellite113	0	3	0	1
Satellite121	3	0	1	0
Satellite122	1	2	0	1
Satellite123	0	3	0	1
Satellite131	3	0	1	0
Satellite132	2	1	1	0
Satellite133	0	3	0	1
Denver	0	0	1	1
Miami	0	0	1	1

Table B.5: Connection Data for Large LEO - Scenario 1.

Node	Cross Links	Up/Down Links
Satellite 111	2	1
Satellite 112	2	1
Satellite 113	2	1
Satellite 121	2	1
Satellite 122	2	1
Satellite 123	2	1
Satellite 131	2	1
Satellite 132	2	1
Satellite 133	2	1
Satellite 141	2	1
Satellite 142	2	1
Satellite 143	2	1
Satellite 151	2	1
Satellite 152	2	1
Satellite 153	2	1
Satellite 161	2	1
Satellite 162	2	1
Satellite 163	2	1
Denver	0	2
Miami	0	2

Table B.6: Connection Data for Large LEO - Scenario 2.

Node	Cross Links	Up/Down Links
Satellite 111	3	1
Satellite 112	3	1
Satellite 113	3	1
Satellite 121	3	1
Satellite 122	3	1
Satellite 123	3	1
Satellite 131	3	1
Satellite 132	3	1
Satellite 133	3	1
Satellite 141	3	1
Satellite 142	3	1
Satellite 143	3	1
Satellite 151	3	1
Satellite 152	3	1
Satellite 153	3	1
Satellite 161	3	1
Satellite 162	3	1
Satellite 163	3	1
Denver	0	2
Miami	0	2

Table B.7: Connection Data for Large LEO - Scenario 3.

Node	Cross Link-A	Cross Link-B	Up/Down Links
Satellite111	3	0	1
Satellite112	2	1	1
Satellite113	0	3	1
Satellite121	3	0	1
Satellite122	1	2	1
Satellite123	0	3	1
Satellite131	3	0	1
Satellite132	2	1	1
Satellite133	0	3	1
Satellite141	3	0	1
Satellite142	1	2	1
Satellite143	0	3	1
Satellite151	3	0	1
Satellite152	2	1	1
Satellite153	0	3	1
Satellite161	3	0	1
Satellite162	1	2	1
Satellite163	0	3	1
Denver	0	0	2
Miami	0	0	2

Table B.8: Connection Data for Large LEO - Scenario 4.

Node	Cross Link-A	Cross Link-B	Down Link-A	Down Link-B
Satellite111	3	0	1	0
Satellite112	2	1	1	0
Satellite113	0	3	0	1
Satellite121	3	0	1	0
Satellite122	1	2	0	1
Satellite123	0	3	0	1
Satellite131	3	0	1	0
Satellite132	2	1	1	0
Satellite133	0	3	0	1
Satellite141	3	0	1	0
Satellite142	1	2	0	1
Satellite143	0	3	0	1
Satellite151	3	0	1	0
Satellite152	2	1	1	0
Satellite153	0	3	0	1
Satellite161	3	0	1	0
Satellite162	1	2	0	1
Satellite163	0	3	0	1
Denver	0	0	1	1
Miami	0	0	1	1

Table B.9: Connection Data for MEO - Scenario 1.

Node	Cross Links	Up/Down Links
Satellite 11	2	1
Satellite 12	2	1
Satellite 13	2	1
Satellite 21	2	1
Satellite 22	2	1
Satellite 23	2	1
Satellite 31	2	1
Satellite 32	2	1
Satellite 33	2	1
Satellite 41	2	1
Satellite 42	2	1
Satellite 43	2	1
Denver	0	2
Miami	0	2

Table B.10: Connection Data for MEO - Scenario 2.

Node	Cross Links	Up/Down Links
Satellite 11	3	1
Satellite 12	3	1
Satellite 13	3	1
Satellite 21	3	1
Satellite 22	3	1
Satellite 23	3	1
Satellite 31	3	1
Satellite 32	3	1
Satellite 33	3	1
Satellite 41	3	1
Satellite 42	3	1
Satellite 43	3	1
Denver	0	2
Miami	0	2

Table B.11: Connection Data for MEO - Scenario 3.

Node	Cross Link-A	Cross Link-B	Up/Down Links
Satellite 11	3	0	1
Satellite 12	2	1	1
Satellite 13	0	3	1
Satellite 21	3	0	1
Satellite 22	1	2	1
Satellite 23	0	3	1
Satellite 31	3	0	1
Satellite 32	2	1	1
Satellite 33	0	3	1
Satellite 41	3	0	1
Satellite 42	1	2	1
Satellite 43	0	3	1
Denver	0	0	2
Miami	0	0	2

Table B.12: Connection Data for MEO - Scenario 4.

Node	Cross Link-A	Cross Link-B	Down Link-A	Down Link-B
Satellite 11	3	0	1	0
Satellite 12	2	1	1	0
Satellite 13	0	3	0	1
Satellite 21	3	0	1	0
Satellite 22	1	2	0	1
Satellite 23	0	3	0	1
Satellite 31	3	0	1	0
Satellite 32	2	1	1	0
Satellite 33	0	3	0	1
Satellite 41	3	0	1	0
Satellite 42	1	2	0	1
Satellite 43	0	3	0	1
Denver	0	0	1	1
Miami	0	0	1	1

Selected Bibliography Including Cited Works

[1] *35th Annual Symposium on Foundations of Computer Science, 20-22 November 1994, Santa Fe, New Mexico, USA*. IEEE, 1994.

[2] Robert E. Bixby, E. Andrew Boyd, and Roger Z. Ríos-Mercado, editors. *Integer Programming and Combinatorial Optimization, 6th International IPCO Conference, Houston, Texas, USA, June 22-24, 1998, Proceedings*, volume 1412 of *Lecture Notes in Computer Science*. Springer, 1998.

[3] J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications.* The Macmillian Press Ltd., fifth edition.

[4] Richard A. Brualdi. *Introductory Combinatorics.* Pearson Education., Upper Saddle River, NJ, 2004.

[5] Robert Carr and R. Ravi. A new bound for the 2-edge connected subgraph problem. In Bixby et al. [2], pages 112–125.

[6] Reinhard Diestel. *Graph Theory.* Springer-Verlag Heidelberg, New York, NY, 2005.

[7] Sándor P. Fekete, Samir Khuller, Monika Klemmstein, Balaji Raghavachari, and Neal E. Young. A network-flow technique for finding low-weight bounded-degree spanning trees. *CoRR*, cs.DS/0205050, 2002.

[8] Richard M. Karp. *Reducibility Among Combinatorial Problems.* Plenum, New York, NY, 1972.

[9] Sven Oliver Krumke, Madhav V. Marathe, Hartmut Noltemeier, R. Ravi, and S. S. Ravi. Approximation algorithms for certain network improvement problems. *J. Comb. Optim.*, 2(3):257–288, 1998.

[10] Madhav V. Marathe, R. Ravi, Ravi Sundaram, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt III. Bicriteria network design problems. *CoRR*, cs.CC/9809103, 1998.

[11] Deepankar Medhi and Karthikeyan Ramasamy. *Network Routing: Algorithms, Protocols, and Architectures.* Morgan Kaufmann Publishers, San Francisco, Ca, 2007.

[12] Deepankar Medhi Michal Pióro. *Routing, Flow, and Capacity Design in Communication and Computer Networks.* Morgan Kaufmann Publishers, San Francisco, Ca, 2004.

[13] R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time (extended abstract). In *FOCS* [1], pages 202–213.

[14] R. Ravi, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1):58–78, 2001.

[15] R. Ravi, Balaji Raghavachari, and Philip N. Klein. Approximation through local optimality: Designing networks with small degree. In Shyamasundar [16], pages 279–290.

[16] R. K. Shyamasundar, editor. *Foundations of Software Technology and Theoretical Computer Science, 12th Conference, New Delhi, India, December 18-20, 1992, Proceedings*, volume 652 of *Lecture Notes in Computer Science*. Springer, 1992.