ECE 527/627

Project Assignment 3
Routing in LEO Satellite Networks

By Peter Pawelski, Alvin Disla Acevedo, Joshua Grassia (Group 11)

1. Formulation of Dynamic Routing in LEO Satellite Networks

- The topology of LEO satellites is time varying.
- Satellites are constantly moving at high speeds and following different paths, therefore cannot be easily tracked in a model.
- Satellites do not stay in the same place for long periods of time, therefore neighboring links or routes may not be valid after a short period of time.
- Constant quick movement affects the quality of the transmission links between satellites and leads to constant handoffs.
- Neighbors will always have the distance between them change

How do frequent link switches and delay impact the adoption of traditional static routing algorithms in LEO networks?

- The static routing algorithm is designed for a fixed topology.
- The static routing solution will work well for snapshots of the satellite network.
- Satellite networks are constantly changing with time
- Frequent link switching/delays change the structure of the network topology.
- Static routing solution would need to be recalculated every time step.
- Satellite links may no longer exist by the time the static routing algorithm comes up with a path solution (due to delay)
- Frequent changes in link will make static routes not valid after short bit of time
- Traditional routing has trouble adapting to frequent and quick changes in the topology.

Discuss why traditional routing algorithms (such as static routing and shortest path routing) are not suitable for LEO satellite networks.

- Static routing and shortest path are built more for static systems.
- Static routing:
    - best suited for small internetworks where the topology and equipment rarely change.
    - Doesn't automatically adapt to changes in a dynamic network because the static routing works off of a manual configuration.
- Shortest path:
    - Doesn't account for other factors in the dynamic topology.
    - Doesn't mean the most reliable or suitable for the network action needed.

Formulate the dynamic routing optimization in LEO satellite networks considering the previous aspects. Explain your approach. When formulating the optimization problem, justify the selection of the utility, variables, and constraints

- The problem: We want to maximize the overall utility of messaging over LEO sat nets.
- Maximize overall utility of sending messages through the satellite network as shown below:

$$\max \sum_{m} U_m(bw_m, d_p)$$

- This represents the sum of utilities for all requests ***m***.
- Utility of each message is dependent on the bandwidth of the message and the path delay.
- $U_m(bw_m, d_p)$ is the utility function for request ***m*** using path ***p***. This function balances the bandwidth requirement (bw_m) with the transmission delay (d_p) using weighting factors.
- The utility of each message should be the maximized based on the bandwidth and delay used to promote overall maximum utility of the network
- A 1 to 1 reward **r** to the algorithm will be provided based on the the utility achieved by each **m** request **r** = $U_m(bw_m, d_p)$
- The bandwidth should be no greater than the minimum available capacity among all links in the chosen path
- Bandwidth constraint represented here:

$$bw \leq \min_{acl \in \mathbf{AC}} acl,$$

- **bw** is the required bandwidth for a specific transmission.
- **AC** is the set of available capacity of links in a selected path **p**
- $min_{acl \in AC} acl$ represents the minimum available capacity among all links **AC** in the selected path **p**.
- A breakdown for calculating the Utility of the message:

$$U_m(bw_m, d_p) = U_{\alpha_1}(bw_m) - \lambda U_{\alpha_2}(d_p)$$

- α1, α1, and λ are constants that are set between 0 to 1. α1 & α2 are usually the same value, and λ determines the importance of the delay.
- Calculating the utility of the bandwidth $U_{\alpha 1}(bw_m)$ and delay $U_{\alpha 2}(d_p)$ both use the following formula (as seen from the GRouting paper):

$$U_\alpha(z) = \frac{z^{1-\alpha}}{1-\alpha}$$

- In other words, to calculate the utility of the bandwidth use:
  - $U_{\alpha 1}(bw_m) = \frac{bw_m^{\,1-\alpha 1}}{1-\alpha 1}$
- And to calculate the utility of the delay:
  - $U_{\alpha 2}(d_p) = \frac{d_p^{\,1-\alpha 2}}{1-\alpha 2}$
- These equations allow us to figure out the utility of the bandwidth and delay which are used to find the Utility of the message.

2. MDP (Markov Decision Process)

GNNs are used to learn a representation of the satellite network, which has a non-Euclidean data structure. This representation allows GNNs to generalize across various satellite network topologies, which means they can handle the time-varying nature of these networks. A DRL is then applied to select the optimal routing path between two satellites based on the representation learned by the GNN. This tries to maximize network resource utilization while staying within the transmission delay requirements.

GNN is a neural network that deals with graph-structured data. The core idea of GNN is that the state of each node in the graph is related to the state of its neighbor nodes.
The general framework of GNN is a message passing neural network (MPNN). This takes inputs of some initial state of the graph and outputs a representation it learned.
This MPNN has two phases: message passing and readout
Message passing consists of the neighbors sending messages to the center node, aggregating all these messages and updating the hidden state of the center node. After this, the readout function outputs a representation (the transition state of each node).

DRL framework includes:
State space, action space and a reward function

- In this case, the state space consists of all possible values of the output representation.
- The action space is all the candidate paths for all possible source to destination pairs in the network
- The reward function indicates the immediate reward provided to the DRL agent after performing a specific action.

The Markov decision process is a 4-tuple that includes:

- State space
- Action Space
- Transition probability
- Immediate Reward

State Space (S)
Consists of all possible values of the graph representation (h_p) learned by the GNN. Each state h_p captures the current topology and link states of the satellite network. The state space is represented as $S = \{h_{p1}, h_{p2}, ... \}$.

The state of the satellite network at any time is represented as a graph $g = (V,E)$, where V represents the set of satellites and E represents inter-satellite links. Each link $e \in E$ has a state vector $e = (f_1, f_2, f_3, f_4, f_5)$, where:

- f_1: available capacity,
- f_2: occupied bandwidth,
- f_3: betweenness value,
- f_4: action vector (indicates link usage),
- f_5: zero padding.

Related to the 5 different fields, there are many states the satellites may encounter. As mentioned before, the satellites' state will be related to the states of its neighbors.

With the graph-like design of the network, we will only focus on the neighbors that are adjacent.
The states of these satellites will be constantly and quickly changing due to their time varying positions.
The state of each satellite would need to be checked periodically.

The graph state representation hp learned by the GNN is aggregated from these link states. Thus, the state space is:

$$S = \{h\_p1, h\_p2, \ldots\},\ h\_p = R(\{ h_K^{\,o} \mid o \in g \}),$$

where R is the readout function after K-iterations of GNN message passing for node o.

Action Space (A)
Comprises the candidate paths (P) calculated for all possible source-destination pairs in the satellite network using the k−shortest−paths algorithm. Each action represents the selection of a specific path for data transmission. The action space is denoted as:
A = {p_1, p_2, ..., p_k}, P = k-shortest-paths(G)

Within the action space, certain paths will be more favorable than others. More favorable in this case would be the path that provides the most utility. The utility would be calculated at each inter link of the proposed path using the current state of the satellites, then total over the whole path followed. However, these paths would also need to follow constraints such as the bandwidth(bw_m) not being greater than the minimum available capacity among all links in the chosen path. This will help prune the number of paths available/possible in the action space while focusing on the paths with maximum utility.

Actions are subject to bandwidth constraints, ensuring:

$$bw_m \leq min_{e \in p}(f\_1(e)),$$

where bw_m is the bandwidth requirement of request m, f_1(e) is the available capacity of a link e.

Transition Probability (T)
With the action chosen for the set of satellite links, neighbors of these satellites will also be affected. With every action taken, the GNN learns optimal paths over time. The GNN will be calculating the total utility of the possible paths until there is little to no increase. In other words, the GNN starts with a training process where eventually the agent's reward based on its optimal path choice converges. Since the GNN learns and develops a better understanding of the topology and its limits, probabilities of choosing certain paths based on certain conditions will be composed. Eventually over time, the network can try certain paths earlier that have a higher chance of providing greater utility. Finding an answer quicker over time is very helpful in a fast moving environment like this. These probabilities will be able to tie in with other conditions in the network.

The transition probability T(s,a,s′) [s: state space, a: action space, s':next space] models how the graph state changes after selecting a path p∈P. This depends on the dynamic environment, including topology updates (due to satellite movement) and link state changes. The GNN iteratively updates its understanding of s′ through message passing:

$M_{k+1}(o) = \sum_{i \in N(o)} m(h_k^i, h_k^o)$, $h_{k+1}^o = U(h_k^o, M_{k+1}(o))$.

Reward Function (r)

Based on the utility function (U_m), which considers the bandwidth requirement (bw_m) of a request (m) and the transmission delay (d_p) of the chosen path (p). It aims to achieve high throughput while maintaining low transmission delay. The reward function is: $r = U_m(bw_m, d_p)$.

In other words, the utility from the path will be the same as our reward function. As the network chooses a path with greater utility, it is the same as providing a greater reward to the DRL agent. The DRL agent is striving to achieve as much utility as it can. In the end, the network will choose the path with the greatest possible utility that fits within the bandwidth constraint.

The reward is based on a utility function $U_m(bw_m, d_p)$ that balances throughput and delay. It is defined as:

$U_m(bw_m, d_p) = U_{\alpha 1}(bw_m) - \lambda U_{\alpha 2}(d_p)$,

where d_p is the delay(in hops) of the selected path, $U_{\alpha}(z) = (z^{1-\alpha})/(1-\alpha)$ is the utility function, and λ is a weight factor. The reward for taking action a in state s is:

$r(s,a) = U_m(bw_m, d_p)$

This pseudocode is how we will tackle the optimization problem for this project. This MDP formulation is fixed on dynamically finding the optimal path of LEO satellites in a constellation.

3. Design of Graph Neural Networks (GNN) for Satellite Network Representation

The satellite network is set up as a directed graph where there is a line of satellites orbiting on similar paths.
Nodes represent individual satellites.
Edges represent inter-satellite links (ISLs) between satellites that are currently within communication range.
As the satellite constellation moves, the edges in the graph change dynamically, reflecting the creation and destruction of ISLs due to orbital motion.

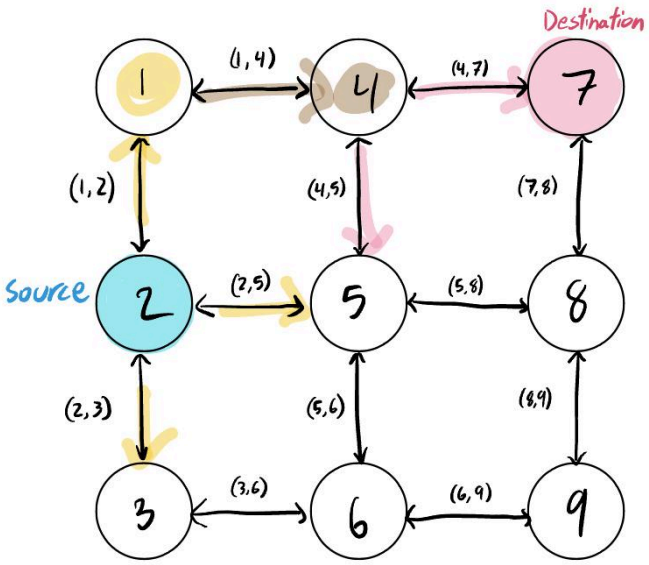
When communicating with gateway stations on Earth, the gateway has access to a satellite and that location is set as a virtual node. Once this initial satellite leaves the virtual node, the next neighboring successive satellite takes its place in this node, performing a handoff. The topology does not change but the link states will be affected. The grid of satellites is all made up of virtual nodes with similar relationships between satellites.

The framework of GNN is a message passing neural network where there are three main phases. The first stage is neighboring nodes sending a message to the center node. The center node will then aggregate or combine as a whole all the message to create a cohesive message.

In each time step, satellites exchange messages containing network state information with their neighboring satellites (adjacent nodes in the graph). This process allows satellites to maintain awareness of their local connectivity and the state of the network. Each node gathers the information received from its neighbors and updates its internal state (node representation) based on this aggregated information.

4. Reinforcement Learning Framework for Routing Optimization

Below is an example set up of the GRouting network layout with randomly generated values:



$$U_m(bw_m, d_p) = U_{\alpha_1}(bw_m) - \lambda U_{\alpha_2}(d_p)$$

$$\alpha_1 = \alpha_2 = 0.5 \qquad \lambda = 0.25$$

$$U_{\alpha_1}(bw_m) = \frac{bw_m^{\,1-\alpha_1}}{1-\alpha_1}, \quad U_{\alpha_2}(d_p) = \frac{d_p^{\,1-\alpha_2}}{1-\alpha_2}$$

alpha1	alpha2	lambda
0.5	0.5	0.25

Node	bw_m	d_p	U(bw_m)	U(d_p)	U(bw_m,d_p)
1	60	31	15.49193338	11.13552873	12.7080512
2	35	25	11.83215957	10	9.332159566
3	50	18	14.14213562	8.485281374	12.02081528
4	40	37	12.64911064	12.16552506	9.607729376
5	50	34	14.14213562	11.66190379	11.22665968
6	35	45	11.83215957	13.41640786	8.4780576
7	60	65	15.49193338	16.1245155	11.46080451
8	50	54	14.14213562	14.69693846	10.46790101
9	80	19	17.88854382	8.717797887	15.70909435

(Utility Calculations with randomly assigned bw_m & d_p values)

Path Name	Nodes in Path	Total Um	Avg Um
A	2,1,4,7	43.10874466	10.77718616
B	2,5,4,7	41.62735313	10.40683828
C	2,5,8,7	42.48752476	10.62188119
D	2,5,6,9,8,7	66.67467671	11.11244612
E	2,3,6,5,4,7	62.12622601	10.354371
F	2,3,6,5,8,7	62.98639764	10.49773294
G	2,3,6,9,8,7	67.46883231	11.24480539
H	2,3,6,9,8,5,4,7	88.30322137	11.03790267

(Utility Calculations of Various Paths)

To solve the MDP, we start off by solving for the utility of each node. Based on this, we can create a shortest path solution from the source to the destination node. But to maximize the overall utility through deep learning, we must find many different random paths and solve for their average utility. Eventually we find a path that maximizes utility over the long run, which becomes our DRL solution.

Ways the GNN assists in the DRL algorithm

- Graph Representation Learning
    - [explained in MDP]
- Generalization Across Topologies
    - Since GNNs operate on graph structures, they can generalize to different network topologies and time-varying states.
- Efficient State Representation
    - Captures key features like congestion, available bandwidth, and link quality, enabling the DRL agent to make informed decisions.
- The GNN provides data that describes the network topology better than typical static layouts. Because of this, the DRL can form a better understanding of the network for more efficient routing.
- Because of the generalized representation of the network topology, the system is a lot more robust to changes in the actual topology.
- GNNs also reduce the training time required for the DRL because of how it compacts the topology data into a generalization.

5. Simulation and Performance Evaluation

Key Parameters
The parameters are chosen to match the values found in the paper. Key details about the code were copied from the paper and pasted into a document titled “Paper Code Features.pdf”. The code created for this report is found in the python file titled “GRouteV5.py” Both files can be found in the same folder as this report.

Constellation
The satellite constellation used in the simulation uses a 11 x 6 two dimensional directional grid to represent the Iridium constellation. This constellation includes 6 orbits with 11 satellites each, totaling 66 satellites/virtual nodes. The distance between each satellite is based on how the satellites are separated in real life. The angle between satellites in the same orbit is 31.6 degrees while the angle between orbits is 22 degrees. The radius of the Earth is 6.378x10^6 m and these satellites orbit at altitudes of 780x10^3m. Using trigonometry, the x and y spacing are found as 2,782,700 m and 4,051,000 m respectively. To make the model less complex, we decided to not include wrapping for the grid.

Traffic Demand
To further simplify the problem, the source and the destination satellites are each chosen randomly from two sets of 6 selected satellites. The selected nodes for the source are the first column of the grid and the selected nodes for the destination are the last column of the grid.

For simulating random packet requests, there is a set of three possible bandwidth requirements: 16,32 and 64 bandwidth units. Per request, one of these values is selected and used. This value will affect the decisions of the agent as it will not hop to nodes with an available capacity lower than this requirement.

Each node in the constellation is provided with constrained random values for signal to noise ratio (SNR) and bandwidth allocation. Under realistic conditions, a satellite can have an SNR ranging between 5dB and 20 dB. With this in mind, each node per request is provided with a random integer value between 5 and 20 for its SNR. This value will then affect the available capacity. Alongside this, a random integer value between 100 and 125 is generated to simulate the nodes experiencing different levels of high traffic from other requests, messages, functionality or environmental conditions.

Network Mobility Models
In this section the network mobility model is based on using a message passing 3 layered fully connected neural network with a Deep Q Network (DQN) agent for training. There were three different actions we chose to have the DQN agent learn from. The first method was to focus on choosing an action (neighbor to hop to) that provided the greatest reward (utility). We compared this method to the DQN agent using the shortest path and random pathing.

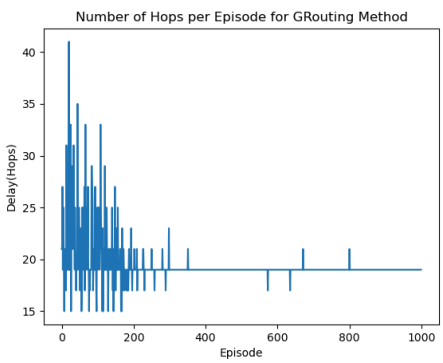
Scalability of the Code
The code provided has the ability to account for larger constellations by updating the variables m and n. These variables control the size of the two dimensional grid used. It also has the ability to increase the traffic demand of the nodes in the network. The range of integer values for the SNR and allocated

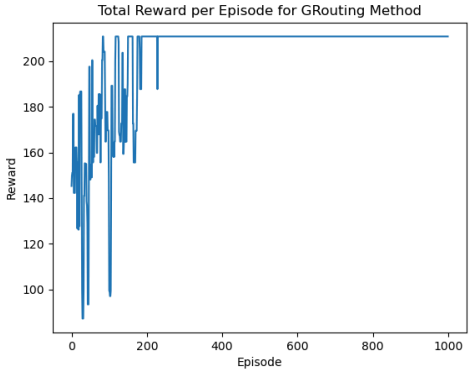
bandwidth can be used with more conservative values. The code should behave in a similar manner to how we simulated it for this report. The values will differ of course but the trends should be related. The hyperparameters may need to be further optimized. Such hyper parameters include the exploration rate, exploration decay, learning pool buffer size and the number of episodes.

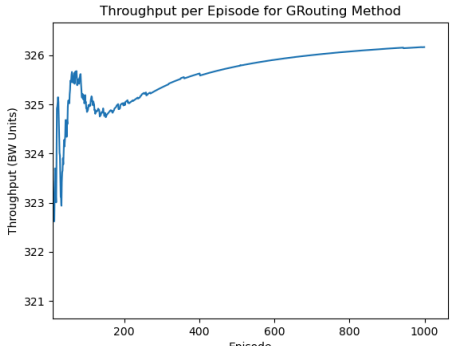
The GRouting framework is scalable due to its use of GNNs and DRL, which help generalize to larger and more dynamic networks. However, computational complexity and real-time constraints will pose a challenge for very large constellations with higher traffic. To deal with this, optimizations can be made such as graph partitioning (divide the satellite constellation into smaller subgraphs) and/or adaptive sampling (reduce size of action or state space evaluated during DRL training and sample the most promising paths).

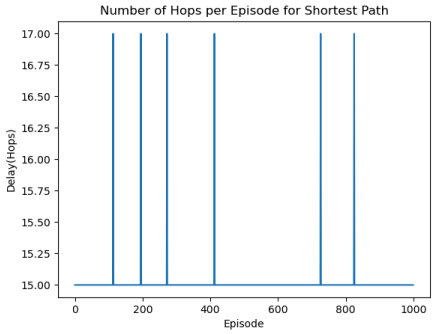
Parallel computing can also be utilized and can be included in this code for future use to simulate higher node demands. Currently, the code only runs one request at a time, however, if multiple instances of the code run on the same topology map, then that would provide more realistic traffic conditions.

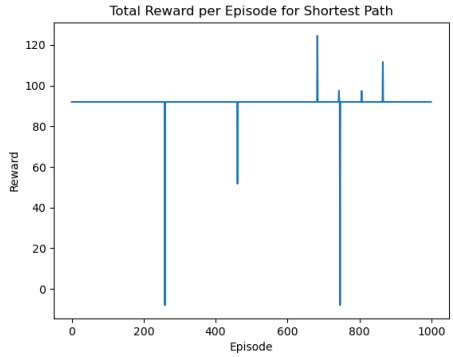
Visual Data of Hops, Rewards, and Throughput over number of Episodes for GRouting, Shortest Path, and Random Path Algorithms

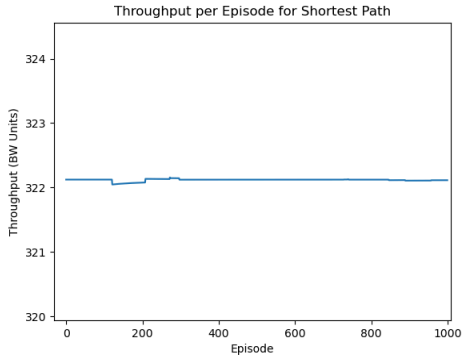


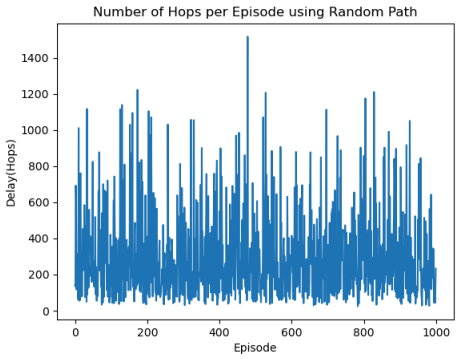


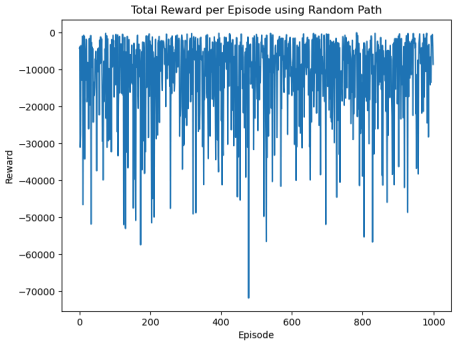


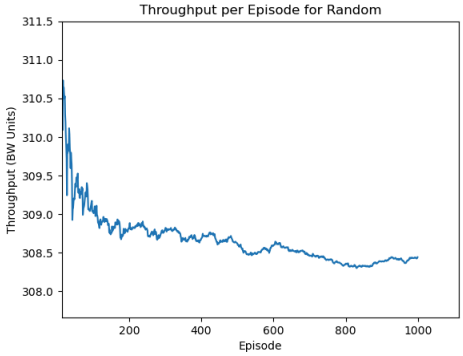












The **GRouting** method was able to converge to an optimal solution at **around 200 episodes**.
GRouting (top row), Shortest Path (middle row) and Random Path (bottom row)
Number of Hops (left column), Total Reward (middle column), Throughput (right column)

Note:
There are spikes in the data due to time when the agent decided to be exploratory and it didn't work out.
Please take note of the scale used for each graph.

Performance

Overall the GRouting algorithm performed very well compared to the other two baseline methods.

Related to the total reward, this particle request performed over 100% better than the shortest path. On average, the random path generated negative reward due to its constant hopping.

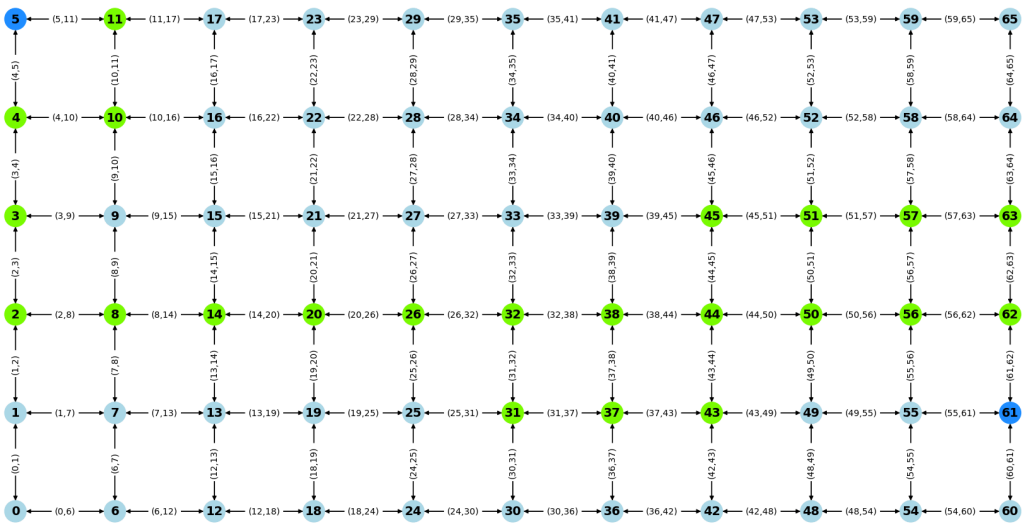
Related to the delay or number of hops, the GRouting method converges to a similar value achieved by the shortest path. The random path method on average hops over 100 times more.

Lastly, we see that the GRouting throughput improves overtime and converges to a value that is 1% higher than the shortest path method. The random path method shows convergence to a lower value that is around 5.5% lower than the GRouting method.

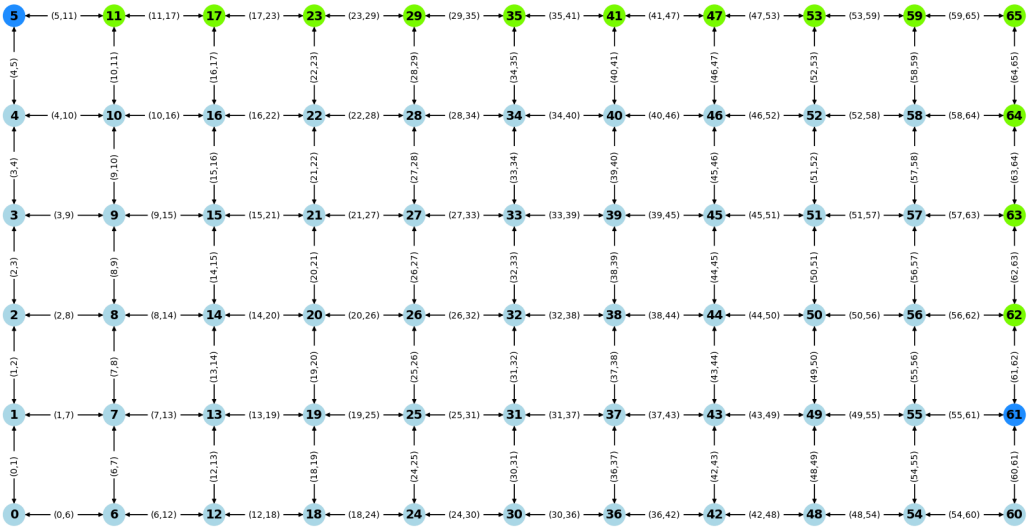
Topological Routes Based on Algorithm

Legend:
Darker Blue = Source or Destination Node
Light Blue = Non-visited Node
Green = Active Node used in Route

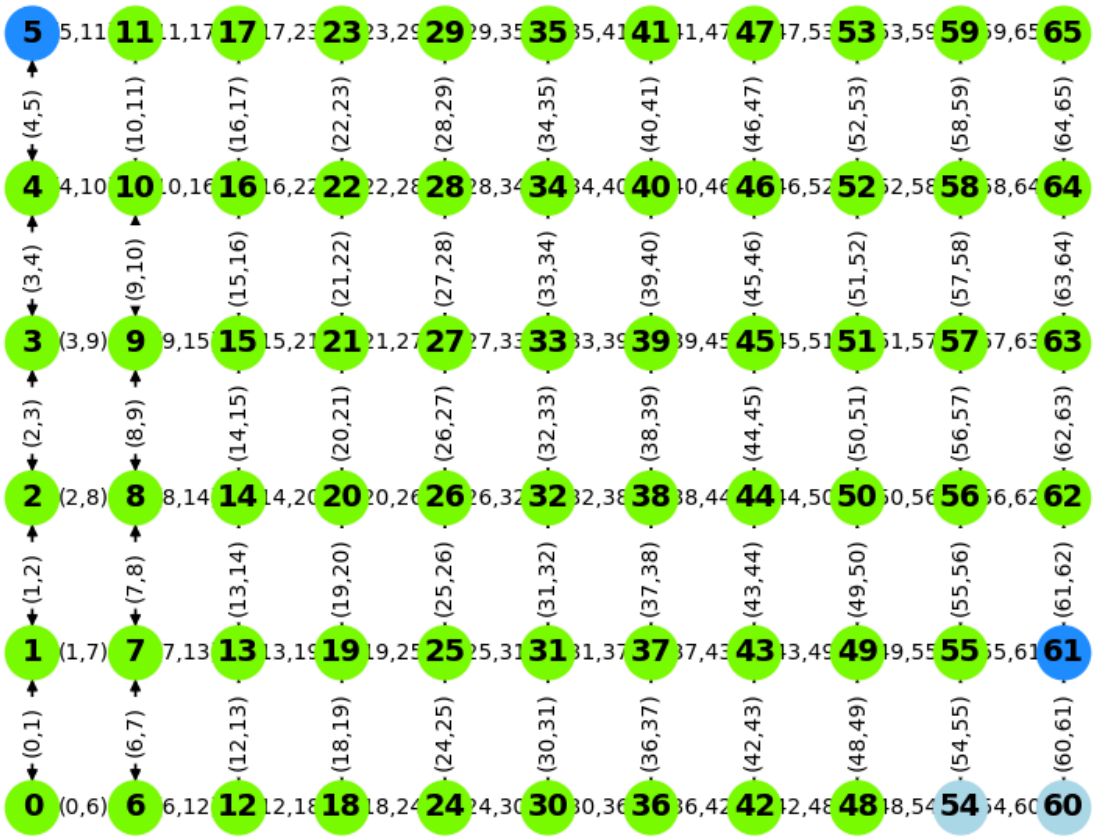
Optimal GRouting Path:



Shortest Path:



Random Path:



*Note that not all of the random paths were this long, but paths like this were not uncommon.