LEO Satellite Network Orchestration with Heterogeneous Graph Neural Networks

Aruna Jayarajan, N. Cameron Matson, and Karthikeyan Sundaresan
School of Electrical and Computer Engineering, Georgia Institute of Technology, USA
{ajayarajan6, ncmatson}@gatech.edu, karthik@ece.gatech.edu

***Abstract*—Low Earth Orbit (LEO) satellite constellations are becoming essential for expanding global Internet access, especially in remote and under-served areas. However, their highly dynamic nature, arising from network mobility, introduces complex coordination challenges between the dynamic satellites and the ground nodes (gateways and terrestrial devices). This is underscored by limited satellite visibility windows and spatially imbalanced user traffic demands. Local association (cell-satellite-gateway) strategies, such as nearest-satellite or greedy load-based selection, result in partial terrestrial coverage or lead to load imbalance that affects traffic demand fulfillment. Network-driven orchestration through centralized optimization can strike an efficient balance between these two key objectives, but is often computationally intensive for periodic operation and real-time deployment. This work presents a learning-based network orchestration framework, `NEO-GNN`, that models a satellite-ground network as a dynamic spatio-temporal graph. In contrast to prior works, it employs a heterogeneous Graph Neural Network (GNN), where satellites, gateways, and ground cells are modeled as distinct node types to capture their varied visibility and networking capabilities. They are trained in an unsupervised manner using tailored loss functions to balance the dual requirements of coverage and utilization, and produce efficient, real-time association decisions during inference. Evaluations show that `NEO-GNN` delivers complete ground-cell coverage, improves traffic demand satisfaction through balanced satellite and gateway use, and remains robust under dynamic visibility and partial satellite failures. `NEO-GNN` provides a scalable and efficient alternative to traditional optimization methods for real-time network orchestration in bent-pipe LEO satellite systems.**

I. Introduction

Low Earth Orbit (LEO) satellite constellations and networks (LSNs) are becoming an integral part of the global internet infrastructure. They consist of thousands of fast-moving satellites designed to deliver broadband connectivity, especially in regions where terrestrial networks are sparse or impractical to deploy. Many current constellations adopt a *bent-pipe* or *single-hop* architecture, where terrestrial user traffic is relayed via satellites to ground-based gateways (GW) without inter-satellite link (ISL) data forwarding. While some modern systems are beginning to incorporate onboard processing capabilities to enable ISLs, bent-pipe designs remain attractive due to their lower latency and hardware simplicity [1].

Regardless, LSN performance is largely bottlenecked by the capacity of their service (cell-satellite) and feeder (satellite-gateway) links (Fig. 1). Hence, to achieve maximum network efficiency of an LSN, it becomes critical to balance the traffic demand carried across satellites and GWs to maximize network capacity/utilization, while simultaneously ensuring coverage of all the terrestrial cells. This challenging dual-objective requires the LSN system to carefully assign satellites to cover ground cells as well as route traffic through available GWs in a dynamic manner that keeps pace with the satellite mobility, while accounting for varying satellite visibility, user traffic demand, and gateway/satellite capacity limits. We refer to this problem as the network orchestration problem in LSNs.

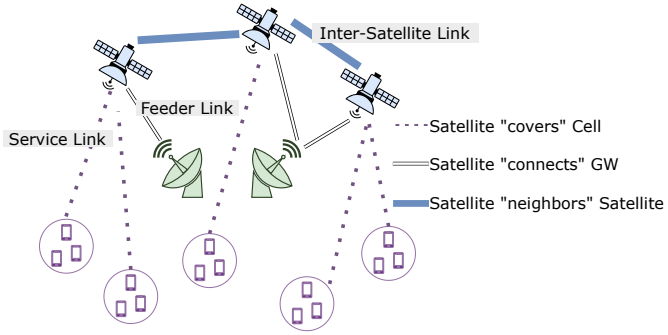


Fig. 1: Heterogeneous Graph for GNN-based Orchestration.

Simple assignment (association) strategies that rely only on local information, such as assigning cells to the nearest satellite or satellites to the least-loaded GW, are fast and easy to implement. However, not having a network-wide perspective often results in poor load distribution and network utilization or incomplete cell coverage by such schemes. On the other hand, advanced solutions that employ global optimization can provide significantly enhanced performance by accounting for the entire network, but are computationally expensive to run in real time. We show it takes over a minute to solve the optimization for a 4750 node network, which is too slow to serve as a centralized controller that must compute frequent orchestration updates on the order of seconds. Thus, we desire an approach that can bring the superior performance of network-driven strategies, but can also scale and operate in real-time for practical deployment in dynamic LSNs.

To this end, we propose a heterogeneous Graph Neural Network (GNN)–based network orchestration framework, called `NEO-GNN`. It operates in real-time to determine configurations (assignments) that strike an efficient balance between satisfying traffic demands (network utilization) and cell coverage even in the face of satellite dynamics. The intuition behind `NEO-GNN` is that LEO satellite orbital trajectories

exhibit spatio-temporal structures that can be learned over time by training on network snapshots that capture the evolving satellite-ground topology. This can help understand how individual satellites should behave in terms of their associations (to cells and GWs) based on where they are so as to balance both utilization and coverage, while respecting visibility and capacity constraints. This work focuses on showing how role-aware graph learning can be applied in practice to large-scale bent-pipe LEO constellations under realistic mobility and capacity constraints. While conventional GNNs for LSNs [2] consider a homogeneous set of nodes, and are unable to cater to both utilization and coverage, NEO−GNN models the LSN as a dynamic, "heterogeneous" graph (like the one shown in Fig. 1) where satellites, ground cells, and GWs are modeled as distinct node types to capture their varied visibility, networking capabilities and constraints, thereby enabling more accurate and flexible decision-making. In addition, NEO−GNN's heterogeneous GNN is trained using loss functions that are tailored to balance the dual requirements of coverage and utilization. By learning satellite association behavior offline, NEO−GNN enables scalable and resilient real-time decisions.

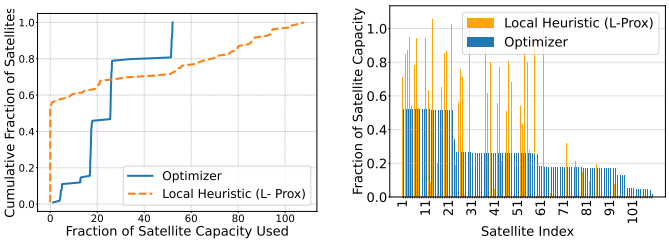
NEO−GNN is evaluated on realistic constellations (similar to those used by Starlink). The results demonstrate that NEO−GNN significantly outperforms both local and network-driven baseline approaches (20–30% more demand satisfaction, 15–25% more coverage) and closely approximates the performance of global optimization-based solutions, in catering to both utilization and coverage. It delivers real-time inference (under 620 ms for a 4750-node network) that scales to larger constellations, unlike optimization approaches, making it well-suited for deployment in dynamic LSN systems. Its unsupervised nature brings resilience, allowing it to sustain performance even during appreciable changes in satellite constellation.

II. Motivation

In terrestrial networks, there are two types network nodes: mobile users and static base stations. The typical cellular deployment make user association and mobility induced handovers simple: users typically are connected to the closest BS and handed over when they move closer to another BS.

LSNs present a fundamentally different paradigm. Rather than two types of nodes, there are three: the user, the satellite, and the satellite GW.

Because of their altitude, the ground coverage area of satellites is much larger than terrestrial cells (even for the relatively low altitude LEO satellites). Even with state-of-the-art high-throughput narrow spot beams [3], the coverage area of several satellites is likely to overlap on Earth. This raises the question: If multiple satellites are visible from a user's vantage point on earth, and multiple GWs are visible from a satellite's location in space, how does the network operator decide the "best" association of user-satellite-GW? Compounding this problem is the fact that satellites, which now represent a key infrastructure node, are highly dynamic, much more mobile than the terrestrial users. The orchestration



(a) CDF (b) Demand plot

Fig. 2: Comparison of local heuristic and optimizer.

problem from a network perspective can be expressed as a two-sided assignment: which ground cells (and all its users) each satellite should serve (service side), and which gateways each satellite should use for backhaul (feeder side). These decisions must be made frequently as satellites move, necessitating an efficient, real-time decision process.

We identify two primary objectives desirable by a network orchestration scheme: 1) Full ground cell coverage, and 2) balanced utilization across the constellation to maximize network capacity. These two objectives must be considered in the face of certain constraints: each satellite (service link) and GW (feeder link) has strict capacity limits that affect the total demand they can serve, and assignments must be made quickly or they risk becoming invalid (e.g. a satellite is no longer visible) due to satellite mobility. Given the likely scenario that total user demand is greater than the capacity of the system, we can either choose to only serve a fraction of users or to serve a fraction of the demand for all users. We now investigate two simple satellite assignment methods (local heuristics and global optimization) and demonstrate how they do a poor job in balancing the objectives given our constraints.

Limitations of Local Heuristics Local assignment methods, such as cells/satellites selecting the nearest satellite or least-loaded GW respectively, are fast and easy to implement. However, they lack awareness of the broader view of the network state and make decisions independently, without accounting for overall system constraints. As a result, this setup frequently causes a few visible satellites or GWs to become overloaded, while others are left idle.

To demonstrate, we have conducted an experiment in which cell-satellite and satellite-GW association is made based exclusively on distance (Details in Section IV). The simulation is based on a constellation comprising 1,584 LEO satellites (similar to the Starlink Phase 1 deployment)[4], 54 ground GWs and 4,569 uniformly distributed user cells throughout the continental U.S. In this baseline model, each ground cell is assigned to the geographically closest satellite. Then, each satellite is paired with its nearest GW to route data to the internet. This heuristic only considers the physical distance (and visibility), not the demand, congestion, or capacity constraints. As illustrated in Figure 2, such local heuristics lead to highly skewed demand distributions, with some satellites exceeding their capacity and many others left idle, showing significant

load imbalance. Even though full coverage is maintained, fewer than half of the satellites are actively serving cells, leading to serious under utilization of the available system capacity.

The problem with this load imbalance is that satellites and GWs have limited capacity, both in terms of physical resources allocated to them [1] and their on-board processing [5]. In the event that the demand assigned to a particular satellite exceeds this capacity, user throughput served can only be a fraction of their requested demand. To illustrate, let the capacity of a satellite be C. Each cell (defined by the set $\mathcal{U}$, indexed by i, $|\mathcal{U}| = N$) generates a demand r_i. Through this local assignment some subset of cells $\mathcal{V} \subset \mathcal{U}$ is assigned to a particular satellite j. The total demand assigned to that satellite is $\tilde{R}_j = \sum_{i \in \mathcal{V}} r_i$. If $\tilde{R}_j > C_j$ the demand served for each cell must be throttled. For fairness, assume each cell is allowed to transmit a fixed "satisfaction" ratio of its desired demand $\lambda \equiv C/\tilde{R}_j$. Obviously, the more demand assigned to the satellite, the lower the satisfaction ratio. Depending on the system, the satisfaction ratio may be the same or unique across satellites and gateways. Regardless, a load balanced system is needed to provide a higher demand satisfaction overall.

Alternatively, rather than allow each user to connect to their closest satellite, the satellites themselves could choose (or be controlled to choose) only as many users at a time not to exceed their capacity. This way the served users would receive their full demand, but many users may receive no resources at all, which is not desirable either.

Why Global Optimization Is Impractical Global optimization methods produce better-balanced solutions by solving large-scale constrained problems. However, they require complete, real-time knowledge of the network state and are computationally expensive. Imagine instead of local decisions, a central controller load balances the entire network, and then broadcasts the results throughout the network. We use the following formulation used for load balancing:

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & L \\
\text{subject to} \quad & \sum_s \sum_j x_{isj}(t) r_j(t) \le \eta L \quad \forall i, \\
& \sum_i \sum_j x_{isj}(t) r_j(t) \le L \quad \forall s, \\
& \sum_i \sum_k x_{isj}(t) = 1 \quad \forall j, \\
& x_{isj} \in \{0,1\} \forall i, s, j
\end{aligned}
$$

where x_{isj} is the binary assignment variable for each flow from user j through satellite s to GW i. The variable η is used to adjust the difference in capacity between satellites and GWs. The objective is to minimize the RHS variable L in the inequality constraint. This is equivalent to minimizing the maximum load on any satellite or GW.

We solve this multidimensional assignment problem[1] to perform load balancing across the network using the commercial solver Gurobi [7]. Unsurprisingly, we see in Figure 2 that the median satellite load is much lower after optimization. However, for a smaller problem than the one considered in the results of Figure 2 with only 600 users and 20 GWs, Gurobi takes 609 s to bound the solution to within 2% of the optimal value. Considering that LEO satellites are visible for a few minutes at most, it is infeasible to run this type of end-to-end, global optimization at the speed of satellite mobility, not to mention the small network size. In Table I we report the solve time of a "quasi-global" optimization on several network scenarios with varying numbers of GWs, cells, and satellites. The quasi global optimizer solves the multidimensional assignment problem in two stages, first optimizing the cell-satellite connection, followed by satellite to GW. Even solving this sub-optimal version of the problem for small networks takes a prohibitive amount of time, making such approaches impractical for real-time deployment.

[1]The multidimensional assignment problem is NP-Hard [6] thus, and furthermore there exist no known polynomial-time approximation algorithms.

TABLE I: Quasi-Global Optimization Run Time

GW	Cells	Satellites	Total Nodes	Opt. Solve Time (s)
15	1147	54	1216	2
21	1511	76	1608	10
54	4569	127	4750	64

Why Learning-Based Models, and GNNs Specifically, Are Suitable LSNs are dynamic, structured, and partially observable. As satellites move along predictable orbital paths, visibility relationships between ground cells, satellites, and GWs evolve over time. Since no single node has full network visibility, assignment decisions must rely on local context and real-time constraints. Learning-based orchestration offers a promising alternative. By training on past visibility and demand patterns, a model can learn to make fast, effective decisions using local and structural features. Graph Neural Networks (GNNs) are particularly well-suited for this task. They work directly on graph-structured data and support localized message passing, allowing satellites to coordinate based on their current neighborhood. A heterogeneous GNN can be employed to represent the distinct roles of satellites, GWs, and ground cells, enabling type-specific message passing and constraint-aware decision-making. The proposed model, NEO−GNN, is built on this architecture and is described in the following section.

III. Design of NEO−GNN

A. Overview

NEO−GNN represents the network as a heterogeneous graph, where each node type—satellite, GW, ground cell—has unique attributes and responsibilities. It uses a Heterogeneous Graph Neural Network (HetGNN) architecture [8], with type-specific message passing and role-aware aggregation, to reason about connectivity and resource allocation. This design enables the model to generalize orchestration strategies not just from a single network snapshot, but across a sequence of dynamic graphs that reflect evolving constellations.

The model is trained without labeled assignments, using constraint-based losses to enforce coverage, capacity, and clear

assignments. While described as unsupervised, the training is fundamentally driven by system feasibility, with penalties for uncovered cells, overloaded satellites, and ambiguous decisions. This is done for two reasons: First, generating a large dataset of optimized labels is impractical given the scale of the problem. Second, the constraint-based formulation allows the system to adapt gracefully under stress, such as gateway congestion or satellite failures, while respecting physical and architectural limits.

The model is trained across many orbital snapshots. This ensures that embeddings capture temporal patterns, such as repeated passes and coverage gaps—and enables generalization to unseen scenarios. The result is a scalable, real-time orchestration approach that improves load balancing, gateway utilization, and coverage fairness.

NEO−GNN is designed to quickly approximate the globally optimal cell–satellite–gateway assignment (over 100× faster than direct optimization), while effectively balancing the coverage–utilization trade-off that challenges local heuristics.

In the following sections we first give a formal definition of the problem and underlying graph structure (Section III-B), then we provide details of the GNN model at the heart of NEO−GNN (Section III-C), describe the multi-objective loss function used and the training methodology (Section III-D), and finally we discuss deployment considerations (Section III-E).

B. Problem and Graph Definition

Efficient orchestration in Low Earth Orbit (LEO) satellite networks presents a unique challenge due to the dynamic topology of the system, mobility of the infrastructure, and strict architectural constraints. As satellites move along their orbital trajectories, the visibility to ground cells and gateways changes frequently, complicating the task of maintaining continuous user coverage and achieving effective load distribution and traffic demand satisfaction. The core problem is to assign each satellite a feasible subset of ground cells to serve and an appropriate gateway (or another satellite) for data egress. These assignments must satisfy several real-world constraints:

- **Visibility constraints:** Satellite visibility is determined by the elevation angle of the satellite as observed from the ground. A satellite is considered visible only when its position is sufficiently above the horizon, exceeding a minimum elevation threshold that ensures reliable line-of-sight communication.
- **Capacity limits:** Satellites and GWs are subject to finite throughput limits, restricting the amount of traffic they can handle.
- **Architectural constraints:** Satellite topologies may be either bent-pipe or support ISLs which can increase the flexibility in assignment.

As defined in Sec. II each cell has a total desired traffic rate/demand, each satellite and GW has a corresponding capacity. In the event that the total demand assigned to a satellite or GW exceeds that capacity, all of the users assigned to that node experience a drop in throughput proportional to the overload.

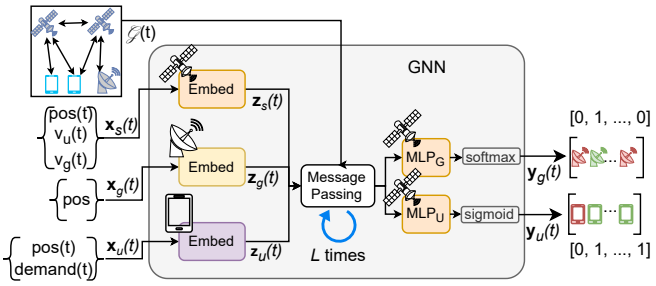


Fig. 3: Forward Pass Through NEO−GNN GNN

Though satellites are always moving, we take the common approach (and backed by observed Starlink behavior [9] and discretize time into small snapshots, in this case 20s intervals. At each time index t, the satellite network is represented as a typed graph:

$$\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t), \quad \mathcal{V}_t = \mathcal{S}_t \cup \mathcal{C} \cup \mathcal{B}, \quad \mathcal{E}_t = \mathcal{E}_t^{\text{vis}} \cup \mathcal{E}_t^{\text{ISL}}$$

Where $\mathcal{S}_t$, $\mathcal{C}$ and $\mathcal{B}$ are the sets of satellite nodes (dynamic), cells, and GWs respectively. $\mathcal{E}_t^{\text{ISL}}$ denotes logical satellite–satellite edges used only for message passing; our evaluation focuses on bent-pipe architectures without ISL data forwarding, while the edge subset $\mathcal{E}_t^{\text{GSL}}$ is visibility-based edges connecting satellites to cells and GWs, A GSL edge exists at time index t (i.e. $e_t^{ij} \in \mathcal{E}_t^{\text{GSL}}$) if satellite j is visible from ground node i in the interval $[t, t+1)$.

C. GNN Architecture

The general structure of NEO−GNN's GNN model is shown in Figure 3. Graph Neural Networks (GNNs) have been widely applied to satellite problems such as routing [10], [11], scheduling [12], and handover management [13]. However, these models generally rely on *homogeneous* graph formulations, where all nodes and edges are treated the same. This overlooks the distinct roles and constraints of each network entity, where ground cells act as passive demand sources, gateways provide fixed-capacity backhaul, and satellites operate as mobile relays making assignment decisions under visibility constraints. Such a distinction becomes critical, when catering to the dual objective of coverage and utilization.

In contrast, Heterogeneous GNNs are designed to model graphs with multiple node and edge types, allowing role-specific processing and type-aware message passing. Although they have been explored in satellite resource scheduling [14], their use for real-time orchestration under dynamic visibility, mobility, and architectural constraints remains limited. The model presented in this work uses: typed **nodes** with separate embeddings for satellites, gateways, and ground cells. typed **edges** with separate message functions for satellite–cell, satellite–gateway, and satellite–satellite interactions, and **role-aware aggregation** allowing satellites to aggregate messages differently depending on the neighbor type.

The GNN works as follows: Given the graph $\mathcal{G}_t$ for a specific snapshot of the network, for each node $v \in \mathcal{V}_t$ we have a time-dependent feature vector $\mathbf{x}_v(t)$. The contents of

each feature vector depends on the specific role: For cells the feature vector, $\mathbf{x}_u$, is position and its time varying demand; for GWs, $\mathbf{x}_g$ is position. and for satellites the feature vector, $\mathbf{x}_s$, include the time-varying position as well as multi-hot encoded vectors indicating the visibility to cells and GWs. These raw feature vectors are fed into node-specific learnable **embedding**. Each embedding projects the raw feature vectors into a 64 dimensional latent space as $\mathbf{z}_u$, $\mathbf{z}_g$, and $\mathbf{z}_s$, respectively.

The next phase of the forward pass through the GNN is "message-passing" described in Eq 1.

$$\mathbf{z}_{s,i}^{(l+1)} = \sigma\Big(W_{\text{self}}\mathbf{z}_{s,i}^{(l)} + W_{\text{cell}} \sum_{k\in\mathcal{N}_{\text{cell}}(i)} \mathbf{z}_{u,i} + W_{\text{gw}} \sum_{m\in\mathcal{N}_{\text{gw}}(i)} \mathbf{z}_{g,m} + W_{\text{sat}} \sum_{j\in\mathcal{N}_{\text{sat}}(i)} \mathbf{z}_{s,j}^{(l)}\Big) \quad (1)$$

Message passing is equivalent to the convolution operation in CNNs in that its an aggregation of information among "nearby" nodes; however, whereas in CNNs the notion of nearby is restricted to fixed filter geometries, in GNNs the neighborhood is defined by the graph structure. The message passing procedure is repeated multiple times which allows information to flow from multi-hop neighbors to each node. In each message passing round, the embedding of a satellite i, $\mathbf{z}_{s,i}$ is updated by aggregating the embeddings of itself, its neighboring satellites ($\mathbf{z}_{s,j} \quad \forall j, e_{i,j} \in \mathcal{E}_t^{\text{ISL}}$), and visible cells and GWs. Each of these is then multiplied by a unique, learnable, square (to preserve the dimension) weight matrix W. The sum of these projections is then passed through a non-linear activation function σ (e.g. ReLU).

After L, rounds of message passing the final embeddings ($z_s^{(L)}$) of each satellite node are fed in parallel multi-layer perceptrons (MLPs) and non-linear activation function to obtain the estimated cell coverage list and target node (either GW or another satellite if no GWs are visible). The output shape of MLP_U is equal to the number of cells, and the non-linear activation is sigmoid, i.e. the output vector for each satellite node $\mathbf{y}_g$ is a list of soft probabilities that the satellite will serve each cell. Similarly, the output shape of MLP_G is equal to the combined total of the number of GWs and available feeder satellites (i.e. the number of satellites which are GW visible). Since each satellite may only connect to one target, off-loading node, the activation is softmax. Note that in the event of a bent-pipe deployment, we simply restrict the output to the number of GWs.

D. Constraint-based Loss and Training

1) Losses: The training objective is built to reflect real-world constraints directly within the learning process. Rather than relying on separate optimization after inference, the model learns to make practical and efficient assignment decisions by minimizing a weighted combination of several loss components. The losses fall into three main categories: (i) *Coverage Assurance Losses*, which ensure that all ground cells receive service and all gateways are actively used; (ii) *Capacity Constraint Losses*, which prevent any satellite or gateway from being overloaded; and (iii) *Unique Assignment Losses*, which push the model toward feasible decisions, like choosing just one gateway per satellite or assigning each ground cell to a single satellite.

Each part of the loss function reflects a specific operational need, and together they help the model make assignment decisions that are practical, efficient, and well-balanced. The following notation is used:

TABLE II: Summary of Notations

Symbol	Meaning
N_s	Number of satellites
N_c	Number of ground cells
N_g	Number of gateways
p_{ij}^{cell}	Probability satellite i serves cell j
v_{ij}^{cell}	Visibility: satellite i to cell j (binary)
p_{ik}^{gw}	Probability satellite i uses gateway k
v_{ik}^{gw}	Visibility: satellite i to gateway k (binary)
d_j	Traffic demand for cell j
C_{max}	Maximum satellite capacity
G_{max}	Maximum gateway backhaul capacity

Coverage Losses These terms ensure all key entities in the network, ground cells and gateways, are actively served.

The coverage loss for cells is given in Eq. 2. A similar loss $\mathcal{L}_{\text{gw-cov}}$ exists for GWs as well. This loss encourages *at least one* satellite to cover each cell. The individual loss for each cell will be 0 if the total probability over all visible satellites is at least 1.

$$\mathcal{L}_{\text{cell-cov}} = \frac{1}{N_c}\sum_{j=1}^{N_c} \text{ReLU}\left(1 - \sum_{i=1}^{N_s} p_{ij}^{\text{cell}} \cdot v_{ij}^{\text{cell}}\right) \quad (2)$$

Capacity Constraint Losses To encourage load balancing, we introduce a set of losses which penalize assigning excessive demand to a particular satellite or GW. Let the load assigned to satellite i be:

$$L_i^{\text{sat}} = \sum_{j=1}^{N_c} p_{ij}^{\text{cell}} \cdot v_{ij}^{\text{cell}} \cdot d_j \quad (3)$$

The gateway and satellite capacity losses are then defined as:

$$\mathcal{L}_{\text{capacity}} = \frac{1}{N_s}\sum_{i=1}^{N_s} \text{ReLU}\left(\sum_{j=1}^{N_c} L_i^{\text{sat}} - C_{\text{max}}\right) \quad (4)$$

$$\mathcal{L}_{\text{gw-cap}} = \frac{1}{N_g}\sum_{k=1}^{N_g} \text{ReLU}\left(\sum_{i=1}^{N_s} p_{ik}^{\text{gw}} \cdot v_{ik}^{\text{gw}} \cdot L_i^{\text{sat}} - G_{\text{max}}\right) \quad (5)$$

As with the coverage loss, the ReLU zeros out the loss if our constraint is met. Note that the coverage losses and load balancing losses work in opposition: we could guarantee coverage by covering all cells with all visible satellites, but that would result in too much load assigned to each satellite.

Unique Assignment Losses These losses ensure that each satellite selects a single gateway and each ground cell is assigned to one satellite, producing unambiguous decisions suitable for real deployments.

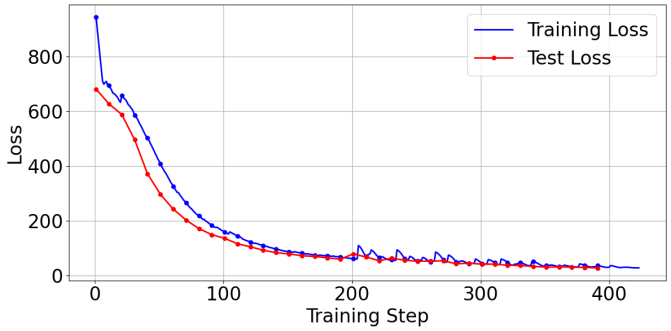


Fig. 4: Total training and testing loss across time-varying graph snapshots.

Satellite-Gateway Assignment Exclusivity Loss To simplify deployment, each satellite typically uses a single gateway within each snapshot. This loss penalizes fractional assignments across multiple gateways, encouraging clear, one-to-one connections for practical routing and scheduling.

$$\mathcal{L}_{\text{sat-assign}} = \frac{1}{N_s} \sum_{i=1}^{N_s} \left| \sum_{k=1}^{N_g} p_{ik}^{\text{gw}} \cdot v_{ik}^{\text{gw}} - 1 \right| \tag{6}$$

Cell-Satellite Assignment Loss Similarly, we assume each ground cell is served by only one satellite at a time. This maximizes coverage and load balancing by allowing satellites to focus on smaller groups of cells and simplifies the access for users.

$$\mathcal{L}_{\text{cell-sat}} = \frac{1}{N_c} \sum_{j=1}^{N_c} \text{CrossEntropy}\left(p_j^{\text{cell}},\ \arg\max_i p_{ij}^{\text{cell}}\right) \tag{7}$$

Total Loss Objective The overall loss is a weighted sum of all individual constraint terms. The weights are tuned during training and selected based on performance and stability: $\mathcal{L}_{\text{total}} = \sum_i \gamma_i \mathcal{L}_i$ The loss weights encode operator trade-offs between coverage and utilization and can be tuned to match service objectives.

Note that the loss used to train NEO−GNN is unsupervised in the sense that there are no ground truth labels. This is for two reasons: The first is that it is difficult to generate a large dataset of "optimal" orchestration decisions given the time required to solve the multi-dimensional assignment problem. The second is that it allows us to incorporate the constraints directly into the losses. Training ML models with constraints is known to be a difficult challenge [15]. Incorporating them into a weighted loss function is a useful method in this case. We should note that the output of NEO−GNN may still violate some of the constraints such as the capacity constraint. This may be inevitable given the ratio of the total demand to the capacity of the system, but in this case the system applies a demand satisfaction ratio as described in Section II.

2) Training Across Time-Varying Graph Snapshots: To reflect the dynamic nature of LEO satellite networks, the model is trained using a series of graph snapshots collected over multiple orbital cycles. Each snapshot captures a unique network configuration caused by satellite movement and evolving visibility to ground cells and gateways. We train on a schedule in two parts. In the first part we train on consecutive snapshots, capturing temporal continuity. Then in the second part, we introduce variability by training on random, non-continuous samples from a different part of the day. We find that training with a batch size of one is most effective. Each epoch consists of multiple training steps on a single snapshot. Figure 4 shows the loss curve on a small dataset of 40 training samples (20 each of continuous and random) with 20 training steps per epoch. The random phase begins at step 200. The steady decline in loss reflects the NEO−GNN's growing ability to generate valid, load-balanced assignments that satisfy coverage and capacity constraints. During the continuous phase, we have a smooth loss curve due to using consecutive snapshots; this is because not much changes between training steps. When we get to the random phase, each new sample results in a small increase in the loss function before adapting. This is expected as these samples represent potentially different satellite configurations and demand patterns; however, these spikes are minor, decreasing, and do not alter the learning trend in the curve.

E. Deployment Considerations

The trained GNN orchestration model can be deployed in the following configurations. They highlight some level of autonomy and operational control in LEO constellations.

Ground-Based Centralized Inference A centralized controller on the ground computes satellite-to-cell and satellite-to-gateway assignments using the full network state per second. It may transmit the updated decisions at a configurable frequency (10-20 seconds). This setup aligns with bent-pipe architectures and requires no onboard processing, enabling globally optimized coordination.

Partial Onboard Inference To extend the autonomy and reduce the dependency on ground based controllers, the next step is to enable satellites to execute a cached version of the GNN model locally. Each satellite recomputes its assignments using local visibility and recent states, allowing it to adapt during ground link loss, gateway overload, or partial network failure. This model does not require full decentralization.

A central question still exists: when do we push the model to satellites, and how often does that need to happen? Typically, this is a one-time deployment, with infrequent updates unless the network undergoes major structural changes. As shown in Section IV, minor demand shifts or localized failures can be handled through local adaptation without retraining.

However, issues like decreased coverage, rising assignment failures, or divergence between predicted and actual load distributions may prompt fallback to onboard models or updated deployments from the ground. Operators usually make this decision based on real-time telemetry. Partial onboard inference enables a hybrid coordination model: leveraging centralized updates when possible, and relying on local inference during

outages. In the future, ISLs may enable lightweight inter-satellite coordination, improving resilience under failures.

IV. Evaluation

A. Experimental Setup

All simulations are conducted over some or all of the continental United States using realistic deployment parameters. User demands are not simulated individually, but are instead aggregated into cells. Cells are defined by Uber's H3 geospatial indexing system. The system uses 4,569 hexagonal ground cells, each covering approximately 26 km^2 on average. Each ground cell generates average of 20 Mbps, and for training, each satellite is assigned a maximum service capacity of 3 Gbps. A uniform demand model isolates orchestration effects from traffic variability, ensuring that load imbalance and coverage gaps reflect coordination behavior. The framework is agnostic to the demand model, with demand used only as a node feature. The 54 gateway locations are based on crowd-sourced reports of real Starlink gateway locations [16]. The simulated satellite constellation is based on Starlink Phase I, consisting of 1,584 satellites arranged across 72 orbital planes, each containing 22 satellites. Although the full constellation is modeled, only the subset of satellites visible over United States at any given timestep, typically around 120, are active satellites for North America. These orbits are inclined at 53° and operate at an altitude of 550 km, yielding an average orbital period of 95 minutes. Visibility between satellites and ground entities (cells or gateways) is determined by geometric line-of-sight constraints and a minimum elevation angle threshold of 25°, reflecting practical antenna steering and propagation constraints. The training dataset consists of 24 hours of 20 second snapshots. Of these 4320 snapshots, a disjoint subset is used for training and testing. Both training and test sets consist of 50% continuous samples and 50% random samples. All results presented (IV), for Optimizer, Local heuristics, Network based models and GNN variants, are obtained from the same test set to ensure consistency and comparability across the models. Though NEO−GNN works for general LSN deployments, for the purpose of evaluation, we only consider bent-pipe connections. This means we only consider satellites which can see a GW.

Model Variants and Baselines: We evaluate the proposed heterogeneous GNN-based coordination model, referred to as NEO−GNN, against four representative baselines:

L-Prox (Proximity Heuristic): Each ground cell is assigned to the geographically closest visible satellite, without considering capacity constraints. Likewise, all satellites connect to their closes visible GW.

L-Demand (Demand-Greedy): Each ground cell is processed in order of highest demand, and assigned to the nearest satellite with available capacity. Satellites are scored using a distance-to-demand ratio, adjusted by current load. GW assignment is the same as L-Prox.

Supervised ML (ML): A static learning model trained on visibility and geographic features.

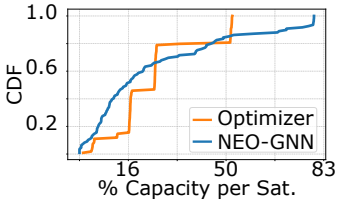


Fig. 5: $\%C$ per Sat. CDF

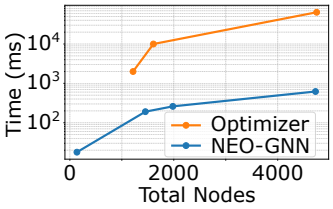


Fig. 6: Runtime vs. Size

Homogeneous GNN (HGNN): A graph-based model treating all nodes identically, without role-aware reasoning [2].

Centralized Optimizer (Opt): A global assignment solver with full access to network state and constraints, used as an idealized performance upper bound.

B. Performance: NEO−GNN vs. Optimized Orchestration

This section compares the performance of NEO−GNN against a centralized optimizer (Opt) that solves the satellite-to-cell and satellite-to-gateway assignment problem as a global combinatorial optimization described in Section II.

Figure 5 shows a CDF of the percentage of satellite capacity used following the orchestration of Opt and NEO−GNN. Both systems are able to limit overloading of satellite nodes through proper load balancing, though NEO−GNN has a bit more variance. As shown in the next section, local and naive ML baseline fail to balance load effectively, resulting in demand satisfaction dropping. Note that both NEO−GNN and Opt guarantee full coverage for the default scenario, where there are sufficient satellites. Because they achieve 100% coverage and all satellites loads are less than the capacity, we also achieve 100% demand satisfaction.

Where NEO−GNN greatly outperforms Opt is during deployment. Figure 6 shows the run time for different sizes of of deployments in terms of the total number of nodes (see Table I for specific node breakdowns; note that the y-axis is in log scale). While the optimization solve time quickly grows to many 10s of seconds, NEO−GNN's inference time remains on the order of milliseconds. This is critical because the orchestration decisions must then be broadcast to the satellites for execution and operation.

Despite slightly higher load variance, NEO−GNN matches the Opt on coverage and utilization while delivering orders-of-magnitude faster inference, enabling rapid reconfiguration in dynamic LEO environments.

C. Baseline Comparison: NEO−GNN vs. Local Heuristics

This section evaluates NEO−GNN against two local heuristics: L-Prox (nearest-assignment) and L-Demand (greedy demand selection). These models are computationally simple, but lack network coordination and structural awareness.

Demand Scaling Evaluation: Figure 7 shows system behavior as total demand increases from 0.6× to 2.0× over the training baseline demand. NEO−GNN consistently maintains over 99% coverage and 95% demand satisfaction across all levels. In contrast, while L-Prox maintains 100% coverage

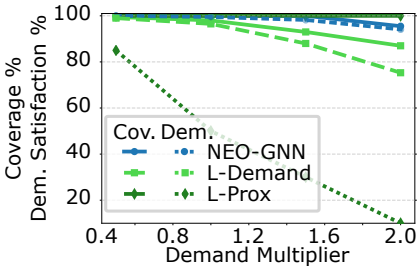


Fig. 7: Coverage (Cov) and Demand Satisfaction (Dem.) vs. demand level

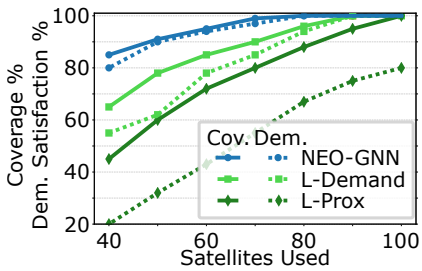


Fig. 8: Coverage (Cov) and Demand Satisfaction (Dem.) % vs. # Sats.

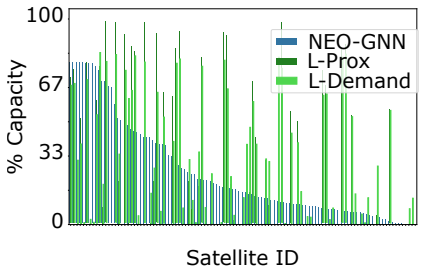


Fig. 9: Per-satellite total demand

(since the orchestration is demand independent) it and L-Demand degrade sharply beyond the $1.2\times$ demand threshold, with coverage gaps and unsatisfied demand increasing rapidly. Unlike the abrupt drops observed in the local heuristics, NEO−GNN demonstrates smooth, predictable degradation under overload, highlighting its robustness. This confirms that the structure-aware coordination logic in NEO−GNN enables better resource sharing and avoids premature load saturation. In contrast, local decisions in L-Demand and L-Prox, made without awareness of broader network conditions, leads to imbalanced load distribution and coverage degradation.

Satellite Scaling Evaluation: We assess this by selecting subsets of satellites from the full constellation and treating only those as active during training and evaluation. For each subset size, training is done using graph snapshots containing only the corresponding satellites, along with visible ground cells and gateways. Figure 8 shows the resulting coverage and demand satisfaction for NEO−GNN, L-Demand, and L-Prox.

As satellite count increases, NEO−GNN consistently delivers superior performance across both coverage and demand satisfaction. NEO−GNN reaches near-100% coverage starting around 50 satellites and maintains it across all larger configurations. L-Demand and L-Prox trail behind, with coverage gaps persisting even at 80+ satellites. NEO−GNN also satisfies a notably higher fraction of the overall traffic demand NEO−GNN continues to scale effectively, maximizing utilization.

Load Distribution Comparison: To understand why, we look at the satellite distribution for a single snapshot. Figure 9 shows the total assigned demand per satellite under the three models. All models serve comparable total demand, however, NEO−GNN distributes it significantly more evenly across available satellites. In contrast, both L-Demand and L-Prox exhibit a higher variance, with many satellites either overloaded or left idle due to uncoordinated, local decisions. This balanced load distribution explains NEO−GNN's superior performance in earlier evaluations in demand and satellite scaling where it sustains higher coverage and demand satisfaction by efficiently utilizing all visible satellite capacity. NEO−GNN's structure-aware inference enables efficient assignment by incorporating local topology and peer activity through message passing, which balances demand across the constellation, prevents local congestion, and maintains full coverage under constraints. These results confirm that NEO−GNN generalizes well to expanded constellations and effectively leverages additional satellite capacity, outperforming myopic heuristics through its scalable and adaptive coordination strategy.

D. Network-Based Model Comparison: ML, HGNN

This section evaluates NEO−GNN against two network-based learning approaches: ML (supervised learning) and HGNN (homogeneous GNN). While both the models incorporate learned inference and consider a global picture, they lack role-specific reasoning and exhibit limited adaptability under dynamic network conditions.

Demand Scaling Evaluation: Figure 10 shows how model performance evolves as traffic demand scales from $0.5\times$ to $2.0\times$ the training baseline demand. As with the local heuristics, these network models also typically degrade in performance with increasing demand, despite being aware of global information. The supervised learning model, ML, exhibits the worst performance in both coverage and demand satisfaction, illustrating the difficulty in using labels to train such a complex optimization problem. On the other hand, HGNN has as good or slightly better coverage than NEO−GNN as the demand increases, but it comes at the cost of lower demand satisfaction which falls to less than 80% at $2\times$ the total load. These results highlight NEO−GNN's superior generalization and robustness under traffic load. Unlike HGNN and ML, which fail to enforce constraints under stress, NEO−GNN maintains performance by balancing load and adhering to satellite visibility and capacity limits.

Satellite Scaling Evaluation: Figure 11 shows how each model performs as satellite count increases. With respect to coverage, NEO−GNN reaches near-100% coverage around 50 satellites and sustains it thereafter. HGNN improves to 90% coerage but saturates, while ML stays below 85% coverage regardless of satellite count. With respect to demand, NEO−GNN maintains over 95% satisfaction throughout. While HGNN peaks near 85% and flattens, ML remains under 75% satisfaction, unable to adjust to capacity growth. Thus, NEO−GNN reaches optimal coverage and satisfaction with as few as 50 satellites and maintains high performance across the full range. In contrast, while HGNN shows moderate improvement but plateaus early, ML fails to capitalize on increased capacity due to its static inference logic.

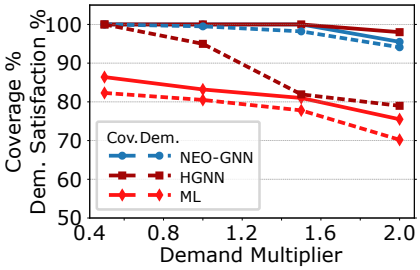


Fig. 10: Coverage (Cov) and Satisfaction (Dem.) % vs. demand level

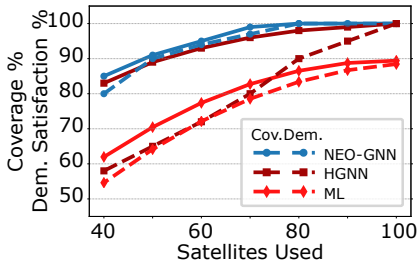


Fig. 11: Coverage (Cov) and Satisfaction (Dem.) % vs. # Sats.

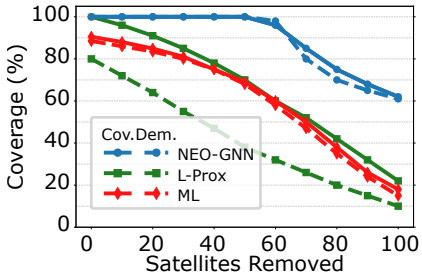


Fig. 12: Increasing satellite failures

E. Resilience to Infrastructure Adaptation

NEO−GNN leverages GNN features to remain resilient to changes in graph topology. Since the graph $\mathcal{G}_t$ is an *input* during inference time, we can handle updates to the satellite constellation or ground segment deployment *after* we have trained NEO−GNN. Figure 12 shows that NEO−GNN continues to perform well even after up to 50% of the satellites *failure* (e.g. for satellites going offline). In contrast, ML and L-Prox degrade rapidly, failing to maintain coverage or satisfy demand under failure. The degradation in NEO−GNN remains smooth, without abrupt collapse, indicating effective load redistribution. This highlights its ability to maintain service continuity under dynamic conditions. Specifically, NEO−GNN sustains over 90% coverage even with 40% of satellite lost. ML and L-Prox drop rapidly, falling below 60% at 50% removal. NEO−GNN also retains over 80% satisfaction after 50% satellite failure. ML and L-Prox drop to 50% or less, failing to recover under loss. NEO−GNN is robust because of its ability to redistribute demand efficiently among the remaining satellites while respecting system constraints.

NEO−GNN also exhibits strong scalability under infrastructure expansion. Even when the active satellite pool increases by 25% during deployment (not considered in training), the model maintains consistent coverage, balanced assignments, and adherence to system constraints,without retraining or reconfiguration. This shows that the learned coordination policy scales to larger constellations and handles added capacity without instability or oversubscription. Overall, the results show that NEO−GNN is resilient to partial infrastructure failures and scalable to future growth, offering fault-tolerant and forward-compatible orchestration in dynamic LEO networks.

V. Related Work

LEO-based NTNs are a cornerstone of 5G/6G evolution, particularly in regions lacking terrestrial infrastructure [1]. Much of LSN satellite research has dealt with load-balancing and routingin *within* the satellite constellation [17], [18], [19]. These works do not consider the end-to-end problem; the traffic matrix already exists. In this work we handle both the ingress and egress part of the problem as well. Machine learning methods have been explored for routing and load balancing, but they generally struggle to enforce constraints and ensure full coverage under dynamic conditions [20], [21].

Graph Neural Networks (GNNs) have been applied to satellite routing, scheduling, and handover management [10], [11], [12], [13], but most assume homogeneous node behavior, limiting their ability to capture role-specific constraints in mixed-role systems. Heterogeneous GNNs (HetGNNs) are designed to model graphs with multiple node and edge types, and they have demonstrated success across domains such as recommendation understanding [8], [22] and region-level geospatial embedding [23]. Nevertheless, their potential in LEO resource coordination—where distinct roles define system constraints—remains largely untapped. Alternative methods such as Ant Colony Optimization (ACO) [19], Federated Learning [5], demand-aware beam hopping and power control [20], and segment routing have been proposed for satellite networks. Each method offers valuable insights, they struggle to simultaneously address the utilization, coverage, and real-time constraints central to LEO network orchestration.

VI. Conclusion

This paper introduces a novel heterogeneous Graph Neural Network (GNN) based learning framework, NEO−GNN for LEO network orchestration. By modeling the network as a dynamic, typed graph, our GNN captures the distinct roles of satellites, ground cells, and gateways. This enables type-specific message passing and constraint-aware inference that allows NEO−GNN to efficiently balance traffic demand satisfaction and cell coverage simultaneously, a key feature that is hard to deliver by existing schemes. Our evaluation demonstrates that NEO−GNN achieves near-optimal performance, ensuring full coverage, balanced load distribution to yield high network utilization, while offering significant computational advantages over centralized optimization, make it suitable for practical, real-time deployments.

Acknowledgment

This work was supported in part by NSF (CNS 2208761).

REFERENCES

[1] S. Liu, Z. Gao, Y. Wu, D. W. Kwan Ng, X. Gao, K.-K. Wong, S. Chatzinotas, and B. Ottersten, "LEO Satellite Constellations for 5G and Beyond: How Will They Reshape Vertical Domains?" *IEEE Communications Magazine*, vol. 59, no. 7, pp. 30–36, Jul. 2021.

[2] Y. Chen, H. Cao, L. Wang, D. Chen, Z. Liu, Y. Zhou, and J. Shi, "Deep Reinforcement Learning-Based Routing Method for Low Earth Orbit Mega-Constellation Satellite Networks with Service Function Constraints," *Sensors*, vol. 25, no. 4, p. 1232, Jan. 2025.

[3] E. Kim, I. P. Roberts, and J. G. Andrews, "Downlink Analysis and Evaluation of Multi-Beam LEO Satellite Communication in Shadowed Rician Channels," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 2, pp. 2061–2075, Feb. 2024.

[4] FCC, "Application for Fixed Satellite Service by Space Exploration Holdings, LLC [SAT-MOD-20200417-00037]," Tech. Rep., 2020.

[5] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "On-Board Federated Learning for Dense LEO Constellations," Nov. 2021.

[6] D. Pisinger, "Where are the hard knapsack problems?" *Computers and Operations Research*, vol. 32, no. 9, pp. 2271–2284, Sep. 2005.

[7] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2026.

[8] C. Shi, "Heterogeneous Graph Neural Networks," in *Graph Neural Networks: Foundations, Frontiers, and Applications*, L. Wu, P. Cui, J. Pei, and L. Zhao, Eds. Singapore: Springer Nature, 2022, pp. 351–369.

[9] H. B. Tanveer, M. Puchol, R. Singh, A. Bianchi, and R. Nithyanand, "Making Sense of Constellations: Methodologies for Understanding Starlink's Scheduling Algorithms," in *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT 2023. New York, NY, USA: Association for Computing Machinery, Dec. 2023, pp. 37–43.

[10] R. Gao, B. Zhang, Q. Zhang, and Z. Yang, "Topology-Compressed Data Delivery in Large-Scale Heterogeneous Satellite Networks: An Age-Driven Spatial-Temporal Graph Neural Network Approach," *IEEE Transactions on Mobile Computing*, vol. 24, no. 07, pp. 6673–6687, Jul. 2025.

[11] M. Liu, J. Li, and H. Lu, "Routing in Small Satellite Networks: A GNN-based Learning Approach," Aug. 2021.

[12] G. Casadesus-Vila, J.-A. Ruiz-de-Azua, and E. Alarcon, "Toward Autonomous Cooperation in Heterogeneous Nanosatellite Constellations Using Dynamic Graph Neural Networks," Mar. 2024.

[13] J.-W. Lee, B. Lim, K.-H. Kim, J.-M. Lee, Y.-S. Ha, Y.-J. Han, and Y.-C. Ko, "Handover strategy for LEO satellite communication using graph neural network," *ICT Express*, vol. 11, no. 2, pp. 239–244, Apr. 2025.

[14] Z. Wang, X. Hu, H. Ma, and W. Xia, "Learning multi-satellite scheduling policy with heterogeneous graph neural network," *Advances in Space Research*, vol. 73, no. 6, pp. 2921–2938, Mar. 2024.

[15] A. Cotter, H. Jiang, M. Gupta, S. Wang, T. Narayan, S. You, and K. Sridharan, "Optimization with Non-Differentiable Constraints with Applications to Fairness, Recall, Churn, and Other Goals," *Journal of Machine Learning Research*, vol. 20, no. 172, pp. 1–59, 2019.

[16] "Starlink Ground Station Locations (2025)," Jan. 2023.

[17] C. Han, W. Xiong, and R. Yu, "Load-Balancing Routing for LEO Satellite Network with Distributed Hops-Based Back-Pressure Strategy," *Sensors*, vol. 23, no. 24, p. 9789, Jan. 2023.

[18] T. Liu, C. Sun, and Y. Zhang, "Load Balancing Routing Algorithm of Low-Orbit Communication Satellite Network Traffic Based on Machine Learning," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, p. 3234390, 2021.

[19] R. Zhi, J. Wang, and Z. Xu, "Load Balancing Routing Algorithm for LEO Satellite Networks Based on Ant Colony Optimization," *Internet Technology Letters*, vol. 8, no. 3, Apr. 2025.

[20] R. Zhao, J. Cai, J. Luo, J. Gao, and Y. Ran, "Demand-Aware Beam Hopping and Power Allocation for Load Balancing in Digital Twin empowered LEO Satellite Networks," Oct. 2024.

[21] Z. Ding, H. Liu, F. Tian, Z. Yang, and N. Wang, "Fast-Convergence Reinforcement Learning for Routing in LEO Satellite Networks," *Sensors*, vol. 23, no. 11, p. 5180, Jan. 2023.

[22] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous Graph Neural Network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 793–803.

[23] X. Zou, J. Huang, X. Hao, Y. Yang, H. Wen, Y. Yan, C. Huang, and Y. Liang, "Learning Geospatial Region Embedding with Heterogeneous Graph," May 2024.