SaTE: Low-Latency Traffic Engineering for Satellite Networks

Hao Wu
National University of Singapore
Singapore
hao_wu@nus.edu.sg

Yizhan Han
National University of Singapore
Singapore
hanyizhan@u.nus.edu

Mohit Rajpal
National University of Singapore
Singapore
mohitr@u.nus.edu

Qizhen Zhang
University of Toronto
Toronto, Canada
qz@cs.toronto.edu

Jingxian Wang
National University of Singapore
Singapore
wang@nus.edu.sg

Abstract

This paper explores traffic engineering (TE) for large-scale Low-Earth-Orbit satellite constellations. While there is rich prior work on TE algorithms for global cloud wide-area networks (WANs), they are designed for static network topologies and often require significant computation time for large-scale networks. Such limitations make existing WAN TE algorithms unsuitable for large-scale satellite networks which rapidly change topology and require computing optimal traffic allocation under stringent latency constraints.

We present SaTE, a low-latency TE algorithm for large-scale satellite networks, computing traffic allocation at millisecond latency. SaTE formulates a heterogeneous graph to model the TE problem, adapting to dynamic satellite topologies. By removing redundant graph relations, SaTE reduces computational latency, allowing the graph to be efficiently learned by a graph neural network that leverages GPUs to rapidly infer traffic allocations. SaTE also exploits the similarity of satellite network topologies and the geospatial distribution of traffic demands to facilitate model training. We evaluate SaTE through extensive data-driven simulation on today's largest satellite constellation, Starlink with 4236 satellites. Our results show over a 23.5% improvement in satisfied demand with an average TE runtime of 17 *ms*, achieving a 2738× speedup compared to commercial solvers.

CCS Concepts

• **Networks → Traffic engineering algorithms**; • **Computing methodologies → Machine learning**.

Keywords

Satellite Network, Traffic Engineering, Machine Learning

ACM Reference Format:
Hao Wu, Yizhan Han, Mohit Rajpal, Qizhen Zhang, and Jingxian Wang. 2025. SaTE: Low-Latency Traffic Engineering for Satellite Networks. In *ACM SIGCOMM 2025 Conference (SIGCOMM '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3718958.3750524

1 Introduction

Satellite networks have experienced rapid growth in recent years [3], providing high-speed internet [66], direct-to-phone services [12], as well as IoT connectivity for applications like global asset tracking [21] and precision agriculture [41]. The scale of these satellite networks has significantly evolved; for example, Starlink, serving 3 million users, has deployed over 6,000 satellites and plans to expand to 12,000 [50].

As constellations grow and traffic demands from users surge, satellite networks must carefully allocate traffic between satellite nodes to enable high-throughput services. Existing traffic allocation methods for satellite networks, such as congestion avoidance [16, 56, 64] and Quality of Service (QoS) constraints [4, 14, 46] based routing, often create congestion and hotspots in the network due to their distributed nature [17, 51]. Similar issues in terrestrial WANs have driven recent advancements in centralized traffic engineering (TE) algorithms [1, 29, 55, 61, 73, 78], which computes traffic allocation along preconfigured paths as a global optimization problem to achieve goals like maximizing overall throughput. While software-defined networking (SDN) for satellites paves the way for applying these TE algorithms to satellite networks [5, 43, 83], the dynamic nature of satellite networks presents challenges to existing TE algorithms.

(1) Frequent Topology Changes: Unlike the static topology of terrestrial WANs, the topology of LEO satellite networks (e.g., Starlink) changes frequently, ranging from seconds to as fast as every 70 ms (Sec. 2.3.1). These changes

make configured network paths quickly obsolete, often outpacing even a few seconds of TE computation time (Sec. 2.3.2), causing considerable throughput degradation. *(2) Traffic Fluctuations:* user-facing network traffic experiences subsecond dynamics [23], while traffic predictions can naturally be erroneous [61]. These factors necessitate real-time reaction from TE systems to handle instantaneous traffic fluctuations. However, existing TE algorithms cannot meet both dynamics. Commercial solvers [24] and heuristic methods [1, 29, 55, 73] exhibit polynomial or higher complexity in TE computation as network scale increases, taking minutes to compute—too slow to keep up with fluctuating traffic. Recent learning-enabled TE solutions provide fast TE computation on static WANs [61, 78], but their trained models are tied to a single topology and require hours of re-training for each change—impractical for the vast number of varying topologies in satellite networks. Thus, there remains a gap for a low-latency TE solution adaptable to fast-changing topologies and dynamic traffic in satellite networks.

We present SaTE, a low-latency TE algorithm designed for dynamic satellite constellations. At the core of SaTE is a novel *satellite TE graph design* that models the entire TE problem, enabling the use of graph neural network (GNN) layers alone to rapidly compute TE and adapt to frequent changes in topology and traffic. In addition, SaTE facilitates its model training by exploiting the structural similarity of satellite topologies and the geospatial distribution of user-facing traffic demands, significantly reducing the size of the training dataset while generalizing well to unseen topologies. Our detailed evaluation on a simulated 4236-nodes Starlink constellation, reveals that SaTE achieves a 2738× improvement in computational latency compared to commercial solvers [24].

SaTE aims to compute TE for large-scale satellite networks with low latency. Existing GNN-based TE [78] relies on graph designs that capture only partial TE components (e.g., link connectivity) and require dense neural networks (DNNs) layers to model essential relations like traffic demands and path associations. However, DNN layers require fixed input dimensions, fail to generalize to unseen topologies, necessitating frequent retraining. To address this, SaTE formulates a heterogeneous graph[1] that models the entire TE problem. This design eliminates the need for DNN layers, allowing GNNs to learn and solve TE entirely. This approach generalizes to unseen topologies and accommodates dynamic changes in key satellite TE components, such as varying input dimensions for network paths and traffic demands. SaTE also reduces computational latency by removing redundant relations from the graph. The simplified graph is then used to build an attention-enabled GNN for traffic allocation.

[1]A heterogeneous graph represents different types of entities and relations.

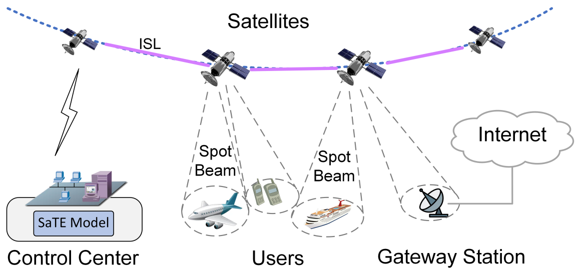


Figure 1: Satellites are interconnected via ISLs and communicate with users through spot beams, providing Internet access via gateway stations. The network states are monitored and managed by a control center, where TE computation is performed. This paper focuses on developing low-latency TE algorithms to compute traffic allocations across end-to-end paths.

Additional considerations in SaTE are training feasibility and effectiveness. For constellations with thousands of nodes, each training data point and its corresponding graph require memory (e.g., 335 GB for Starlink) far exceeding GPU capacity [59]. Moreover, distributing the training of a single graph across multiple GPUs with performance guarantees remains an open challenge [79, 82]. SaTE addresses this issue by pruning non-contributory elements from the TE input, leveraging the sparsity in global distribution of satellite users [7, 49, 69], as satellites over sparsely populated areas (e.g., deserts) often have no traffic demand, leaving many paths idle. This input pruning is enabled by our graph design, which allows the use of GNN layers alone, unlike prior works relying on DNNs that require fixed-sized inputs. Another challenge is the vast number of varying topologies for Starlink. Covering all possible topologies risks overfitting [81] and high training overhead. Instead, SaTE explores the similarity of satellite topologies—while the satellites' geometric positions change as they orbit the Earth, certain network topologies remain similar or isomorphic. Thus, we develop a topology pruning method that reduces the number of training data points required while maintaining SaTE's performance. By combining these pruning methods, SaTE condenses the dataset size, making training feasible while reducing training overhead.

We evaluate the feasibility of SaTE through extensive data-driven simulations on Starlink [18], with 4236 satellites[2]. Our evaluation includes scenarios where Starlink satellites are solely linked with space lasers (as they currently operate with over 10000 lasers [37, 66]) and scenarios where they

[2]Our simulation is based on the four completed orbital shells already launched by SpaceX, for which reliable orbital parameters are publicly available. We chose not to include satellites from incomplete orbital shells, as the lack of a consistent topological structure makes it difficult to model and evaluate those configurations accurately.

are partially linked with ground relays. We generate traffic matrices for 3 million satellite users globally, with fluctuating traffic across services such as voice, video, and file transfer, and varying overall traffic loads (e.g., arrival intensity). The trained model of SaTE is evaluated on unseen topologies and traffic flows. Our results reveal that:

- SaTE achieves 17 *ms* computational latency[3] in solving Starlink's TE problems, improving by 2738× over commercial solvers [24].
- SaTE satisfies at least 11.0% more traffic demand in Starlink, outperforming state-of-the-art methods [24, 35, 55, 78] by over 23.5%.
- SaTE demonstrates generalizability across various scales of satellite networks, tested on unseen topologies and traffic from the trained model.

Contributions: SaTE's core contributions include:

- A low-latency TE computation solution for dynamic large-scale LEO satellite constellations.
- A dataset pruning technique that significantly reduces the training dataset volume, facilitating model training.
- Detailed evaluation of SaTE performance on a simulated mega-constellation, Starlink.

2 Background and Motivation

2.1 Primer on Satellite Networks

This paper focuses on non-geostationary satellite constellations that implement inter-satellite links (ISLs). Examples include Iridium [48] with radio-frequency links, and Starlink using space lasers (up to 200 Gbps [68]) across over 4,000 satellites [37], with some scenarios using "bent-pipe" links [28, 58] (ground relays linking close satellites). Historically, satellite communication relied solely on ground stations; requiring satellites to wait until they were within range of ground stations for data relay. ISLs [28, 34, 53] now enable direct satellite-to-satellite communication, forming an independent switched network—a satellite network. Our work focuses on in-space end-to-end communication scenarios, where traffic is transmitted through multiple satellite hops. Fig. 1 shows a typical satellite network architecture.

Satellite constellations are organized into orbital shells—distinct layers of satellites each operating at specific altitudes. For instance, Starlink's four shells are at altitudes of 540, 550, 560, and 570 km. These shells form the network topology that connects satellites both within the same shell and across different shells. As satellites orbit, the network topology varies, mainly induced by two types of connections:

(1) Links within the same shell: Within each shell, each satellite usually connects with four neighbors: two on the same orbit (intra-orbit links) and two on adjacent orbits (inter-orbit links), as shown in Fig. 2 (a). The intra-orbit links are generally more stable and rarely change, while inter-orbit links are deactivated when satellites reach high latitudes due to the excessive viewing angles between satellites in neighboring orbits [11], leading to changes in network topology. For Starlink, these links operate on space lasers.

[3]The time to compute the optimal traffic allocation.

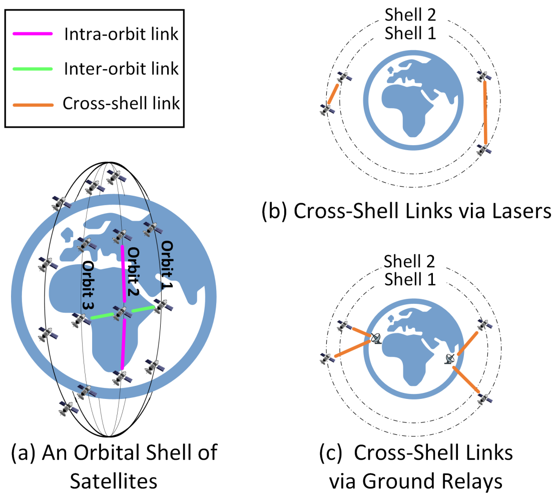

Intra-orbit link
Inter-orbit link
Cross-shell link
Orbit 1
Orbit 2
Orbit 3
(a) An Orbital Shell of Satellites
Shell 2
Shell 1
(b) Cross-Shell Links via Lasers
Shell 2
Shell 1
(c) Cross-Shell Links via Ground Relays

Figure 2: Network topology refers to the physical connectivity structure formed by satellites and links. (a) shows a grid of satellites within a single orbital shell, all at the same altitude, interconnected via ISLs, e.g., lasers. (b) and (c) show a two-shell constellation, with cross-shell links formed by (b) lasers or (c) ground relays using RF.

(2) Cross-shell links: Satellites in different orbital shells can be interconnected via two primary methods, each depicted in Fig. 2 (b & c). The first method involves direct laser links that connect satellites across different orbital shells. These links frequently re-pair due to the relative motion among satellites located in separate shells [44]. The second method connects satellites through ground station relays, known as "bent-pipe" links [47], which are commonly utilized when direct laser systems are not yet fully operational [28, 58]. These ground relay-based radio-frequency links undergo frequent changes as non-geostationary satellites rapidly move relative to the ground.

2.2 Primer on Traffic Engineering

Centralized TE [76] is a well-established concept that has been extensively researched and deployed in production WANs [30]. The TE workflow shown in Fig. 3, involves a control center that periodically gauges traffic demands (by a bandwidth broker [35]), computes traffic allocation, and translates the results into router configurations deployed

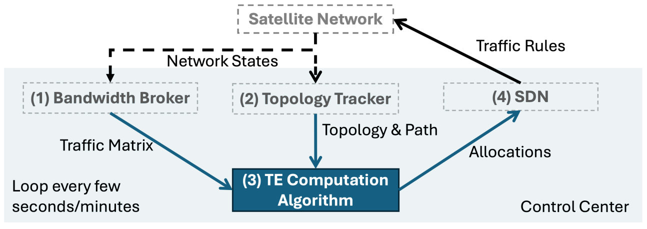


Figure 3: Workflow of Traffic Engineering.

through SDN [6, 19, 36]. Central to TE is an optimization algorithm that computes the allocation of the traffic demand across preconfigured network paths, aiming to optimize a TE objective, such as maximizing overall throughput. We outline a typical satellite TE workflow as follows:

(1) Traffic Matrix Acquisition: Satellite users must establish connections prior to data transmission to facilitate the allocation of channel resources (e.g., beam time slots). This procedure typically involves authentication by the control center, which can also be leveraged to estimate traffic demands based on traffic types and service-level agreements. Examples are provided in Appendix D. The aggregated demands form a traffic matrix, with each entry representing total authorized demand between specific satellite pairs.

(2) Topology Determination: Satellite network topologies change over time but are predefined by satellites' designated orbits. A satellite topology refers to the network's connectivity structure. The TE control center monitors satellite coordinates and network state [32] in real time, adapting to subtle changes in the topology. Thus, the topology is determined in advance of TE computation.

(3) Network Path Preconfiguration: Traffic paths (e.g., from source A to destination B) are precomputed based on the determined network topology. Network operators use routing algorithms and configure these paths with techniques like MPLS labels [22, 55, 78] prior to TE computation.

(3) TE Computation: With the traffic matrix, topology, and paths as inputs, the TE algorithm computes the optimal allocation for each traffic flow across its candidate paths. The TE objective, defined by the operator, typically aims to achieve goals such as maximizing throughput. The mathematical formulation of TE is detailed in Appendix A.

(4) Traffic Rule Distribution and Loading: Traffic allocation results are converted into traffic rules for onboard satellites switches. These rules are distributed to satellites via the control center and propagated through ISLs with minimal delays. Satellites then compile and load the rules into their flow tables.

Scope of The Work: The paper focuses on the TE computation step within the broader TE workflow (Fig. 3). The workflow operates periodically over fixed time intervals, with the duration primarily determined by the runtime of TE computation step. While other steps in the workflow typically take within sub-second timescales[4], traditional TE computation methods [1, 22, 29, 38, 55] require several minutes to even hours for large-scale networks with thousands of nodes. This creates a runtime bottleneck in the workflow, resulting in significant bandwidth waste and delays in traffic allocation adjustments (as discussed in Sec. 2.3.2). Our goal is to develop low-latency TE algorithms designed for large-scale, dynamic satellite networks.

2.3 Dynamics and Implications for TE

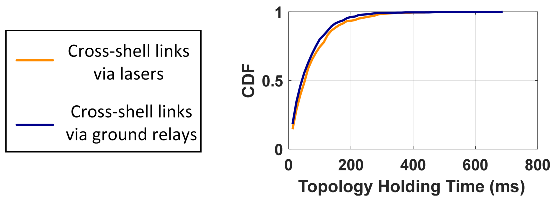
2.3.1 *How Long Does A Topology Hold?*

Changes in network topology are induced by changing ISLs due to the relative motion between satellites and asynchronous movement of the satellites relative to Earth. Here, we provide an analysis of the *Topology Holding Time* (THT), which measures how long a topology remains unchanged. We focus our analysis on Starlink with 4236 satellites.

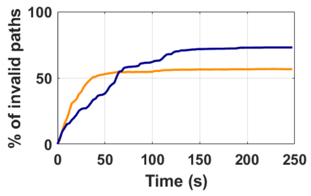
Topology Generation for Starlink: We generate the topology snapshots for Starlink based on FCC specifications [18]. **A)** Each Starlink satellite connects with four neighbors within the same shell using lasers, see Fig. 2 (a). These ISLs remain stable and only break when the satellites reach high latitudes larger than 75°. **B)** Each satellite also connects with the nearest satellite in neighboring shells, forming cross-shell links. Our analysis includes two scenarios with different types of cross-shell links deployed: (B.1) *Cross-shell links via lasers:* Each satellite connects to the nearest satellite in adjacent orbital shells via lasers (see Fig. 2 (b)). The link remains until the distance exceeds 2,000 km. (B.2) *Cross-shell links via ground relays:* Each satellite connects to the nearest ground relay using RF (see Fig. 2 (c)). The link remains until the satellite is at an elevation angle lower than 25° relative to the ground relay [18]. Ground relays are globally deployed at 222 real-world locations [49].

THT Analysis: We sample 40,000 consecutive topology snapshots every 12.5 ms. THT is measured as 12.5k ms, where k is the number of sampled intervals in which the topology remains unchanged. Fig. 4 (a) shows the CDF of THT, with a maximum of around 700 ms and an average of 70 ms. Cross-shell link types do not significantly impact THT.

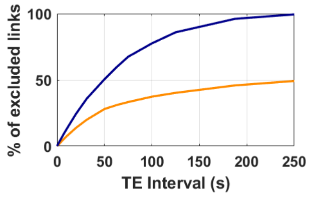
[4]Less than 174 ms for rule distribution (i.e., sending traffic rules from the control center to satellites, as detailed in Appendix D), 56 ms for path calculation (Sec. 4), and approximately 100 ms for satellites to update their flow tables with the received rules [75].



(a) Topology Holding Time



(b) % of Invalid Paths



(c) % of Excluded Links

Figure 4: (a) Average THT for the four-shell Starlink constellation is 70 ms. (b) Configured paths (e.g., shortest paths) become invalid over time due to changes of ISLs. Within 150 seconds, over 56% of 14,941 configured paths in Starlink become obsolete. (c) Longer intervals result in significant link exclusion and waste.

2.3.2 *Implications.*

(1) Implication of Topology Changes: Traditional TE computation for large-scale networks with thousands of nodes often requires several minutes to solve, creating a runtime bottleneck in the periodic TE workflow and delaying subsequent rounds. Then, many configured paths become outdated over time during a TE workflow round, as shown in Fig. 4 (b). To ensure performance, TE computation considers only ISLs that remain consistent during the computation runtime. However, longer computation times result in greater link exclusion and substantial bandwidth waste.

To quantify this impact, we analyze 40,000 consecutive topology snapshots of Starlink sampled every 12.5 ms. For a given interval (i.e., 12.5k ms), we retain only ISLs present in every snapshot. We then calculate the ratio of excluded ISLs to the total number of potentially changing ISLs (a number primarily contributed by cross-shell links), with results shown in Fig. 4 (c) for intervals ranging from 12.5 ms to 250 s. We note that longer intervals result in a significant fraction of ISLs becoming outdated and excluded. This underscores the importance of low-latency TE computation to reduce link exclusion and bandwidth waste.

(2) Implication of Traffic Fluctuations: Networks serve user-facing traffic, which fluctuates over time. These fluctuations prevent pre-allocation of traffic, requiring TE allocation to be computed rapidly in response to received traffic demands. Longer computation times delay adjustments, as traffic demand may shift significantly during the period.

Motivation for Faster TE Computation: Frequent topology changes and fluctuating traffic demands in satellite networks demand low-latency TE algorithms. Faster TE computation can shorten the workflow interval, mitigate the exclusion of ISLs caused by topology changes and enable timely traffic allocation adjustments. This motivates the development of SaTE, designed to compute TE at low latency.

2.4 Applying ML to Satellite TE

Scalability Challenges: Solving TE for satellite networks with thousands of nodes is a large-scale optimization challenge, involving NP-hard linear programming. Existing heuristic methods [1, 22, 29, 38, 55, 71, 73] typically require minutes or even hours to compute solutions. Commercial solvers [24] take 46 seconds for Starlink (see Sec. 5.1), which far exceeds the THT shown in Fig. 4 (a). Consequently, many preconfigured network paths become obsolete due to changes in topologies, leading to degraded network performance.

Acceleration via Machine Learning: ML offers opportunities to overcome TE computation bottlenecks via GPU parallelization. Recent studies [23, 61] use deep neural networks and multi-agent reinforcement learning to accelerate TE in terrestrial WANs but struggle with topology changes in satellite networks, requiring frequent retraining following topology changes. In contrast, model architectures like GNNs are well-suited for learning from graph-structured data with dynamic connectivity (e.g., topologies). Advanced GNNs, like graph attention networks (GATs), are particularly adept at handling variable-sized inputs and generalizing to unseen graphs [9, 72]. This capability presents a promising opportunity to address the challenges of satellite TE.

Limitations of Existing GNN-Based TE Solutions: However, recent GNN-based TE approaches [78] are hamstrung for satellite networks due to their design of the input graph. Specifically, their graphs only represent how individual links (e.g., direct connections between nodes) are used in the configured paths that span source-destination pairs. While this relation can be learned using GNN layers, the graph [78] fails to represent essential TE relations, such as the paths tied to traffic demand. This critical relation is not captured by the graph and instead requires additional learning layers, such as DNN. Consequently, their models, consisting of GNN and DNN layers, introduce the following limitations:

- **Limited Generalization**: As topologies change, the DNN layers trained to process embeddings from a specific set of paths cannot generalize to preconfigured paths in topologies that are not part of the training dataset (referred to as "unseen" topologies). This requires extensive re-training (e.g., 6-10 hours [78]) whenever the topology changes.

- **Memory Challenges:** DNN layers require fixed-sized inputs, which means all preconfigured paths for each source-destination pair must be explicitly represented as inputs to the DNN. In Starlink, with 4,236 satellites and 10 preconfigured paths per source-destination pair, a single DNN input has a path dataset of size 263 GB, exceeding commercial GPU memory. The fixed dimensions prevent the application of flexible pruning techniques, like those developed by SaTE (Sec. 3.4). These limitations make existing learning-based TE approaches struggle to address the scale and dynamics of satellite networks.

3 SaTE – Satellite TE

3.1 Overview

SaTE aims to achieve low-latency TE computation, enabling rapid adaptation to frequent topology changes and fluctuating traffic demands in satellite networks. To achieve this, we transform classic TE optimizations into an inference process of neural networks, which can be accelerated by GPU, leveraging modern deep learning frameworks. The key enabler of SaTE is a graph representation that models the satellite TE problem. The graph is then used to build a GNN framework. To make training on commercial GPUs feasible, SaTE introduces a topology pruning method as well as a traffic and path pruning approach, as shown in Fig. 5. The paper addresses the two main challenges in SaTE's design:

(1) Satellite Graph Design and Learning: Our solution must design a graph that captures the changing interconnectivity in satellite network topologies. Existing approaches [33, 78] construct graphs based solely on the physical interconnectivity, serving as inputs to GNN layers. However, they miss crucial correlations between traffic flows and feasible paths—key for solving TE. To compensate for this, existing methods rely on additional DNN layers, but this combination of GNN and DNN struggles to generalize to dynamic and unseen topologies, requiring frequent retraining. SaTE addresses this by formulating a heterogeneous graph that comprehensively represents all elements and relations critical to the satellite TE problem. This graph enables the use of GNN layers alone, allowing the model to generalize to dynamic changes in the graph (e.g., topology, traffic demands, and paths) and even entirely unseen graphs, thereby alleviating the need for frequent retraining with each topology change. Furthermore, SaTE reduces the number of relations within the graph to remove computational redundancy and enhance inference latency. Sec. 3.2 and 3.3 detail our approach.

(2) Dataset Pruning: The size of the input to the GNN modules increases quadratically with the number of satellites.

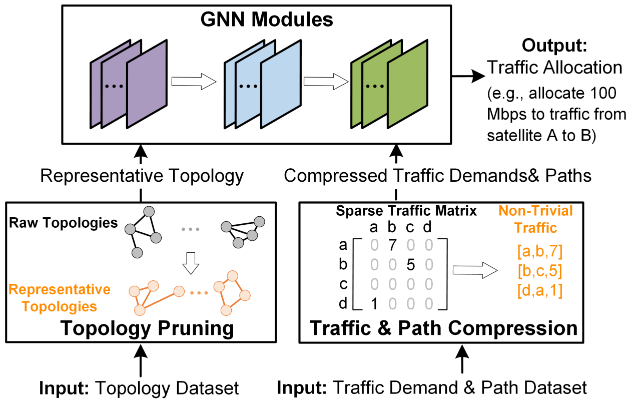


Figure 5: Overview. SaTE applies pruning to topologies, traffic matrices, and paths—all of which constitute the inputs to the GNN. The GNN computes traffic allocation (e.g., distributing 100 Mbps onto preconfigured paths from satellite A to B) to optimize TE objectives, such as maximizing overall throughput.

For example, Starlink, with its 4,236 satellites, generates a data point of 335 GB for TE computation, which exceeds the memory capacity of GPUs [59]. While multi-GPU training is an active field of study, efficiently distributing and computing a singular data point and its corresponding graph across multiple GPUs remains an open question [79, 82]. Instead, SaTE prunes massive input data to make it feasible to train its model on a single GPU. Further, the vast number of varying satellite topologies makes the training challenging as well. Sec. 3.4 addresses these challenges. We note that the pruning step is not merely an optional optimization, but a necessary precondition for scalable training and inference.

3.2 Satellite Graph Modeling

In this section, we describe how SaTE designs a heterogeneous graph representation tailored for satellite networks.

Graph Design: A graph essentially represents a relational mapping (i.e., edges) between elements of interest (i.e., nodes) and serves as the input for any GNN model. Previous works construct graphs to represent computer networks solely from physical interconnectivity and associated preconfigured paths [78], relying on additional DNN layers (e.g., an MLP) that limit their ability to generalize to changing topologies. In contrast, SaTE constructs a heterogeneous *satellite TE graph* to represent the entire TE problem. This graph captures changes in topology, time-varying traffic matrices, and path reconfigurations. These dynamics can then be modeled by GNNs, which leverage their inherent generalization capabilities to adapt even to unseen topologies.

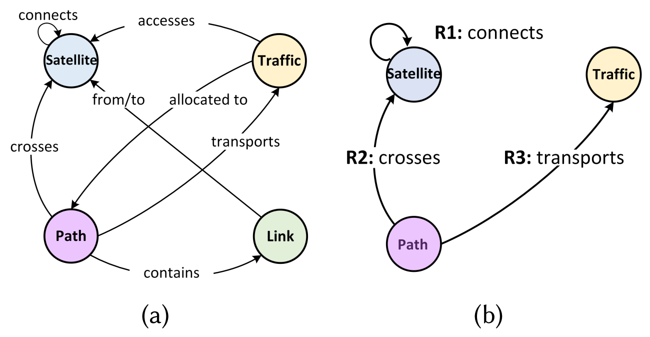


Figure 6: (a) Heterogeneous satellite TE graph. (b) SaTE’s simplified satellite TE graph.

The satellite TE graph includes the following key elements: (1) *Satellite* – denotes the end nodes of ISL, including satellites and potentially ground relays when "bent-pipe" links are used; (2) *Traffic* – denotes traffic demands from users; (3) *Path* – specifies the route taken by traffic from a source to a destination satellite; (4) *Link* – connects two satellites via ISL. We then specify the relations among these elements: e.g., satellites inter-*connect* each other; each path *crosses* chains of satellites from a source to a destination, being able to *transport* network traffic; each path *contains* a group of links. Fig 6 (a) details all possible relations in the graph.

Graph Reduction for Enhanced Computational Latency: GNNs learn complex dependencies and interactions within a graph using message passing [26], where neighboring elements exchange and aggregate information based on specific relations to update their embeddings (i.e., vector representations). However, SaTE’s satellite TE graph contains multiple heterogeneous elements and relations, typically requiring distinct message passing for each relation type (e.g., *connects*, *contains*). As the number of distinct message passing increases, so does the computational latency of the GNN. Therefore, SaTE must simplify its heterogeneous graph.

To address this, SaTE reduces redundancy within the graph by eliminating unnecessary relations while retaining all essential information for TE computation. For example, the "access" relation between *Satellite* and *Traffic* is redundant because it is implicitly covered by the “crosses” and “transports” relations. Specifically, the “crosses” relation connects *Path* to *Satellite*, indicating which satellites are traversed by each preconfigured path. Similarly, the “transports” relation connects *Path* to *Traffic*, capturing the allocation of traffic demands to paths. Together, these relations enable message passing from *Traffic* to *Satellite* via *Path*, ensuring that satellites receive traffic-related information without requiring a direct “access” edge. In addition, the representation of *Link* can be merged into the "connect" relation, with its link incorporated into the weight of the "connect" relation, facilitating the use of edge-weighted attention mechanism [40]. SaTE retains the *Satellite* element due to the dynamic nature of satellite interconnections, which frequently break and reform. Thus, this time-varying relation—where satellites *connect* with each other—must be explicitly modeled and learned by the GNN. Fig. 6 (b) shows the simplified satellite TE graph. This simplification reduces the learning complexity, leaving three types of relational pairs: (R1) Inter-Satellite connections; (R2) Path-Satellite; and (R3) Path-Traffic.

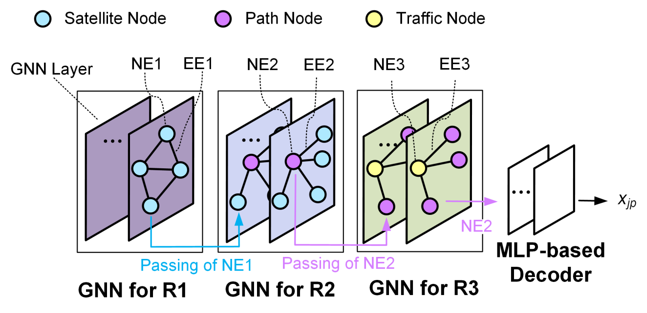
3.3 Learning the Graph

Our discussion so far has covered how SaTE formulates a comprehensive yet simplified satellite TE graph to adapt to changing topologies and other TE elements while reducing computational redundancy. This graph serves as the input for graph neural networks, which are designed to learn representations of graph-structured data [72].

SaTE’s simplified satellite TE graph includes three distinct types of relations (R1, R2, R3, as shown in Fig. 6 (b)). To handle this heterogeneity [70], SaTE uses three GNN modules, each applied to process a type of relation. The architecture for all GNN modules is based on the graph attention network [9], which dynamically prioritizes the importance of neighboring nodes during information aggregation. This allows the neural network to focus on the most relevant connections in the graph, reducing the influence of irrelevant relations. The three GNN modules sequentially aggregate and propagate information according to their respective relations, as shown in Fig. 7. Through this sequential structure, embeddings updated by one module are passed to the next, capturing interdependencies across embeddings and facilitating complex inference through multiple layers of abstraction.

Embedding Initialization: We empirically select the embedding dimension to be 768 for nodes and edges in the simplified satellite TE graph (Fig. 6 (b)). As shown in the table of Fig. 7, each embedding is initialized by multiplying its respective TE input with a 1×768 learnable weight matrix $\mathbf{W}$, randomly initialized at the start. For example, node embeddings of satellites (NE1) are initialized using the number of a satellite’s neighbors; while edge embeddings (e.g., EE1) are initialized with the link capacity. Link or satellite failures are handled by setting the corresponding capacities to zero, which is inherently supported by the GNN design.

Inference through Message Passing: SaTE stacks multiple GNN layers in each module to iteratively update the initialized embeddings. A layer’s input consists of a set of node embeddings $\{v_i \in \mathbf{R}^d | i \in N\}$ where N is the node set and a set of edge embeddings $e_{i,j}$ for all related nodes (e.g., an edge exists between the graph nodes of two connected satellites in the ‘GNN for R1’ module). Each layer computes a weighted



	Definition	Initialization
NE1	Satellite Embedding	$\mathbf{W}_{NE1}$ · #Neighbors
NE2	Path Embedding	$\mathbf{W}_{NE2}$ · Path Length
NE3	Traffic Embedding	$\mathbf{W}_{NE3}$ · Traffic Demand
EE1	Edge Embedding	$\mathbf{W}_{EE1}$ · Link Capacity
EE2		$\mathbf{W}_{EE2}$ · Node's Position
EE3		$\mathbf{W}_{EE3}$ · #Candidate Paths

Figure 7: SaTE's graph is learned by a GNN framework with three modules, each processing a relation shown in Fig. 6 (b). The table summarizes the initialization of node (NE) and edge (EE) embeddings.

average of the embeddings of the neighbor nodes and related edges, followed by a nonlinear function LeakyReLU(·)), to produce a new set of node embeddings $\{v_i' \in \mathbf{R}^{d'} | i \in N\}$:

$$v_i' \leftarrow \Theta_s \cdot v_i + \Big\Vert_{k=1}^{K} \left(\sum_{j \in r(i)} \alpha_{j,i}^k (\Theta_n^k \cdot v_j + \Theta_e^k \cdot e_{j,i}) \right), \qquad (1)$$
$$v_i' \leftarrow \mathrm{LeakyReLU}(v_i'),$$

where α, Θ are learned attention scores and weights, $r(i)$ denotes neighbors of node i, and $\|$ denotes vector concatenation. A detailed description of these parameters is provided in Appendix F. Within the framework, embeddings are updated sequentially across three GNN modules: 'GNN for R1' module computes satellite embeddings. 'GNN for R2' module updates satellite and path embeddings concurrently to mutually enhance their representations. 'GNN for R3' module refines path embeddings and traffic embeddings together. Finally, an MLP decoder computes the traffic allocation x_{jp}, which represents the amount of traffic j assigned to path p.

Correction for Constraint Violation: Neural networks including GNNs are inherently soft constraint models, meaning they approximate solutions but do not strictly enforce hard constraints during optimization. For example, they cannot guarantee that the total bandwidth allocated to a flow will not exceed its demand or that traffic allocation will remain within link capacity limits. As a result, the computation outcome x_{jp} may violate TE constraints. To address this, we trim overloaded traffic, ensuring the final TE solution is feasible. Evaluation results in Sec. 5 account for this trimming.

Training Method: We train the GNN model using supervised learning, with ground-truth labels for traffic allocation generated by the commercial solver Gurobi [24]. These labels are obtained by solving the TE problem with the same set of inputs, including traffic demand, network topology, and paths, ensuring alignment with SaTE's input data. During training, we iteratively calculate the loss between the label x_{jp}^* and the traffic allocation x_{jp} computed by SaTE. The loss is backpropagated to update all model parameters: attention scores α, weights Θ, and embedding initialization matrices $\mathbf{W}$ in each message passing layer. The loss function and hyperparameter are detailed in Appendix B.

3.4 Dataset Dimension Pruning

The input size to the GNN model increases quadratically with the scale of satellite constellations. Mega-constellations like Starlink, with thousands of satellites, result in large-sized inputs. For example, Starlink, with its 4,236 satellites, generates a data point of 335 GB. This includes a 4,236×4,236 float traffic matrix (72 GB), where each element specifies the demand between source-destination pairs, and a 4,236×4,236 path matrix with 10 preconfigured paths per entry (263 GB). This total size far exceeds typical GPU capacity (e.g., 80 GB on an NVIDIA A100 [59]). While multi-GPU training is being explored, training a single data point and its corresponding large graph across multiple GPUs with performance guarantees remains an open challenge [79, 82]. Furthermore, mega-constellations exhibit numerous varying topologies, and training on all possible topologies risks overfitting and incurs high training overhead.

Existing GNN-based TE approaches rely on DNN layers to model key relations to TE [78], but these layers require fixed-sized input dimensions. Furthermore, these DNN layers cannot generalize to dynamic and unseen topologies, requiring frequent retraining. In contrast, in Sec. 3.2 and 3.3, SaTE introduces a graph design that models the entire TE problem and is learned solely by GNN layers. By leveraging attention-enabled GNN [72], SaTE can process variable-sized inputs, enabling traffic and path pruning. Additionally, the inductive learning capabilities of GNNs [9] allow SaTE to generalize to unseen topologies, enabling training on a small set of representative topologies instead of all possible snapshots. This section details two pruning strategies:

Traffic & Path Pruning: SaTE prunes the traffic and path matrices by leveraging the uneven distribution of satellite users [65, 66]. While the entire satellite network provides global coverage, some satellites often receive zero traffic, especially when passing over sparsely populated areas such as deserts [7]. Thus, we retain only the non-zero elements

Network Scale	Path Dataset Volume		Traffic Dataset Volume		Total Reduction
	Original	Pruned	Original	Pruned	
66	0.014	1.1E-04	2.9E-10	7.52E-09	132×
369	0.91	1.9E-03	3.8E-06	3.51E-08	481×
1584	30.45	6.4E-03	4.3E-05	5.57E-08	4751×
4236	335.72	0.015	2.8E-04	1.66E-07	22381×

Table 1: The average volume of one data point (unit: GB). Volume reduction is achieved by pruning.

in the traffic matrix and exclude idle paths with zero traffic input from the training dataset. Such pruning still preserves the spatial dynamics of satellite networks by recording the source and destination of each traffic demand, thus maintaining representational fidelity. Table 1 shows that pruning reduces Starlink data volumes by up to 22,381×, *shrinking a data point from 335 GB to 15 MB*.

The pruning method is applicable only to fully GNN-based frameworks. It cannot be applied to hybrid models with DNNs. This is because DNNs require fixed-size and position-specific input structures, and a model trained on a specific pattern of non-trivial traffic matrix entries cannot generalize to new patterns where the positions of active entries differ.

Topology Pruning : SaTE leverages the fact that certain satellite network topologie snapshots are similar or isomorphic as satellites orbit the Earth. Instead of traversing all possible topology snapshots, we train the model on a small set of representative topologies with diverse structures, where the set size can be customized. Specifically, our empirical study in Sec. 5.3 shows that when trained with 512 representative topologies for Starlink, the model can already outperform state-of-the-art approaches. Appendix E details how we extract those representative topologies from a large set of snapshots using Determinantal Point Process sampling [42].

4 Implementation and Evaluation

SaTE Software: SaTE is implemented in Python, trained and deployed on Microsoft Azure with an NVIDIA A100 GPU, 24 cores, and 220 GB memory. SaTE's model uses traffic matrix, network topology, and ten pre-calculated shortest paths between each source-destination pair to compute optimal traffic allocation. We implemented a satellite trajectory emulator to generate realistic network topologies based on orbital parameters from FCC official specifications [18]. Since Starlink has not disclosed its traffic traces or per-country gateway distribution, we developed an in-house satellite network simulator to produce traffic matrices for evaluation. This work does not raise any ethical issues.

Starlink Constellation: Based on the orbital parameters from FCC specifications [18], we emulate satellite trajectories using the open-source orbital mechanics emulator [63]. The emulation replicates the largest actively operating satellite constellation as of April 2024: Starlink (Phase 1), with 4,236 satellites. These satellites are distributed across four orbital shells, as detailed in Table 4 in Appendix G.

Starlink Topology: We generate the topology snapshots for Starlink based on the emulated satellite trajectories. As of March 2024, Starlink has been operating over 10,000 lasers [67, 68]. Additional details about ISLs, including the conditions under which they form and break, are provided in Sec.2.1 and 2.3.1. Our evaluation presents results for two scenarios with different types of cross-shell links. All links via lasers and ground relay have a capacity limit of 200 Mbps[5].

Other Constellations: To evaluate SaTE's performance on various network scales, we simulate additional constellations: *(1) Iridium:* A constellation of 66 satellites in a single shell at 781 km altitude, using Ka-band ISLs [34]. *(2) Mid-Sized Constellations:* By retaining only the first two orbital shells of Starlink and reducing the number of orbital planes by factors of 8 and 2, we created two mid-sized constellations with 396 and 1584 satellites, serving as intermediaries between Iridium and full-scale Starlink.

Satellite Traffic Matrices: Our simulations place 3 million globally distributed users and 1000 gateways based on population density [20], with adjustments using a smoothing factor to account for remote areas (detailed in Appendix G). Traffic flows include user-to-user services such as voice and video, and gateway-to-user traffic, where users access Internet data through gateways. Traffic is routed directly through satellite links, with users and gateways directly connected to satellites. The traffic parameters are detailed in Table 2.

In each 1-second interval[6], a Poisson process with traffic intensity λ (flows per second) generates new flows between randomly selected user pairs or between a gateway and a user. This process introduces stochastic fluctuations in traffic generated. The parameter λ controls traffic loads. Each flow's traffic demand is associated with the satellites directly connected to the user or gateway and is represented as a demand between the corresponding satellite pair. The traffic matrix is generated by aggregating the total demands of both new and ongoing flows between each satellite pair. Uplink and downlink capacities are set to 50 Mbps per connection.

Path Calculation: 10 shortest paths are precomputed per satellite pair for routing. Instead of recalculating all paths every interval, only those impacted by topology changes—such as newly consistent ISLs or removed links—are updated. This incremental approach detailed in Appendix C updates fewer

[5]We intentionally scaled down both bandwidth and the number of flows proportionally to avoid the overhead of managing the lifetimes of numerous flows. It does not affect the validity of our findings, as our primary focus is on evaluating relative performance trends rather than absolute throughput.
[6]Iridium uses 20-second intervals due to much slower topology changes.

Business Type	Demand	Duration
Voice	64 Kbps[a]	1 to 10 minutes
Video	8 Mbps[b]	5 to 30 minutes
File Transfer	50 Mbps	26 to 130 minutes[c]

a. G.711 standard b. Typical bitrate of 1080P videos
c. Duration corresponds to file transfers of 10 to 50 GB

Table 2: Traffic Flow Parameters

than 2% of paths per second, with an average computation time of 56 ms.

Training and Testing Datasets: We train separate models for different satellite constellations. For each constellation, we collect 10,000 time-varying topologies at an 1-second interval as satellites move and associate them with fluctuating network flows at various levels of traffic intensity. We maintain a training to testing data size ratio of 4:1. For all experiments, the testing dataset consists of completely unseen topologies and traffic matrices from the trained model. After applying traffic and path pruning and topology pruning (Sec. 3.4) on the initial training dataset, we train SaTE on a reduced set of representative topologies and their corresponding traffic matrices.

Objectives and Baselines: Our default objective is to maximize total throughput, with Appendix H.2 focusing on minimizing maximum link utilization (MLU). We compare SaTE against six competing schemes: (1) *Gurobi* [24]: a commercial solver widely used in TE computation, (2) *POP* [55]: a resource allocation method that decomposes a network into fractions with $1/k$ capacities and execute sub-allocation with a part of flows in each fraction for faster TE. (3) *ECMP with Water Filling* [35]: a scheme equally allocates traffic to shortest paths with equal hops until the paths are saturated. (4) *Satellite Routing* [56]: a backpressure-based routing approach for satellite networks and other intermittently connected networks [51, 64]. We compare only performance for this scheme, not computational latency, since its computation is distributed across the routers rather than on a centralized controller. (5) *Teal* [78]: a GNN-based TE method known for its fast inference performance in WANs. (6) *HARP* [2]: a GNN-based TE method for changing topologies.

Performance Metrics: We focus on two critical metrics:

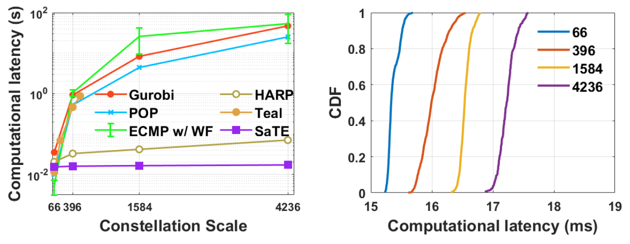
- *Computational Latency:* The time required to compute the optimal traffic allocation for a given network topology, paths, and traffic matrix.
- *Satisfied Demand:* The percentage of total demand met (throughput divided by total traffic demand). We measure satisfied demand in online settings, accounting for TE computation delays, where the current traffic allocation remains in effect until the new one is computed. This online metric highlights the impact of computational latency on real-world network performance.

5 Results

5.1 Computational Latency vs. Scales

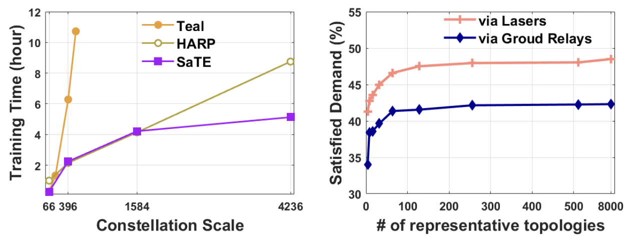
Method: In this experiment, we evaluate the computational latency versus the number of satellites, considering time-varying topologies and fluctuating traffic demands with varying intensities. Our experiments cover scales with up to 4,236 satellites, the largest actively operating constellation as of April 2024. For SaTE and learning-based baselines Teal [78] and HARP [2], we train a separate model for each constellation scale. Other heuristic baselines are directly applied. We note that Teal [78] cannot fit into GPU memory when scaling to Starlink, as it requires over 335 GB per data point. Due to its graph design, our proposed dataset pruning method (§3.4) cannot be applied. Therefore, we compare SaTE with Teal at scales of 66, 176, 396, and 528 satellites.

Results: Fig. 8 (a) shows the computational latency vs. various constellation scales: Iridium (66 satellites), Mid-Size 1 (396), Mid-Size 2 (1584), and Starlink (4236). As expected, the latency of the baselines increases significantly with the number of satellites, while SaTE maintains low and nearly constant inference latency across scales. This is because SaTE's graph design makes computational latency primarily dependent on the number of heterogeneous graph relations—reduced in Sec. 3.2—and the number of GNN layers. In contrast, Teal's latency increases with scale because its input cannot be pruned, as its fully-connected neural network design requires traffic and paths between all source-destination pairs, incurring significant computational overhead. Harp—dedicated to MLU minimization under far fewer constraints than SaTE's throughput-maximizing problem—suffers from 4× higher latency than SaTE because its edge-path embedding transformer's computational complexity scales with network size. In addition, HARP is not inherently adaptable to throughput maximization. Other baselines suffer from high algorithmic complexities, leading to significantly longer latency as the network scales up. Overall, SaTE achieves the lowest average latency of 17 ms at the largest scale—Starlink—which is 2738×, 1462×, and 1013 − 5230× faster than Gurobi, POP, and ECMP with WF, respectively. For Mid-Size 1, SaTE outperforms Teal by 28×. Fig. 8 (b) shows the CDF of SaTE's computational latency across various scales. The slight increase in computational latency at larger scales is due to model memory expansion and access overhead, a common GNN-related issue [74]. Overall, the low computational latency achieved by SaTE enables effective TE that can adapt to the fast changes in topologies and traffic.



(a) Latency vs. Scales (b) CDF of SaTE’s Latency

Figure 8: (a) On the 4236-satellite Starlink constellation, SaTE achieves 1013 × lower latency compared to the best-performing throughput-maximizing baseline [35]. (b) SaTE achieves an average computational latency of 17 ms on Starlink and a standard deviation of 87 μs.



(a) Training time vs. Scales (b) Representative Topologies

Figure 9: (a) SaTE trains faster than [78] [2], with better scalability as constellation size increases. (b) SaTE’s model converges over 512 representative topolgoies.

5.2 Training Overhead

Method: We evaluate the training time of SaTE across various scales and compare it to the learning-based baselines, Teal and HARP. In this experiment, SaTE and HARP are trained to generalize to changing topologies, while Teal is trained on a single, static topology at scales of 66, 176, 396, and 528 satellites—scales Teal can handle. All of them are trained under the same hardware environment.

Results: Fig. 9 (a) shows the training time vs. constellation scales. SaTE outperforms Teal by 1.06× (0.284 h/0.268 h) in Iridium (66 satellites) and by 2.8× (6.28 h/2.25 h) in Mid-Size 1 (396 satellites). Additionally, SaTE achieves a 1.7× improvement over HARP (8.7 h/5.1 h) in Starlink. Teal’s training time increases rapidly with constellation scale, and its model is limited to a single topology. HARP experiences training slowdowns at scale, as its edge-path embedding transformer exhibits computational complexity that grows with network size. In contrast, SaTE’s training time grows much slower while producing a model that generalizes to changing topologies. This reduced overhead is due to SaTE’s simplified TE graph design and its ability to reduce input data volume, resulting in more effective GPU parallelism.

5.3 Impact of Topology Pruning

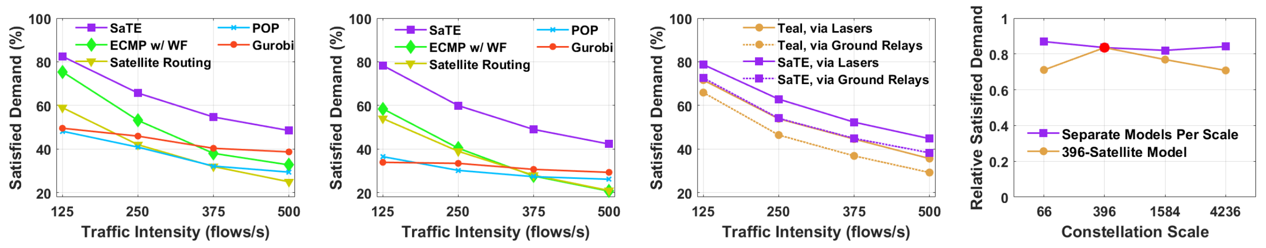
Method: We study the impact of topology pruning by comparing the satisfied demand tested on 2,000 unseen topologies and traffic matrices, using models trained with varying # of representative topologies (up to 512). We evaluate the satisfied demand achieved by the trained model on the 4236-satellite Starlink under high traffic intensity (500 flows/s).

Results: Fig. 9 (b) shows the satisfied demand as we train SaTE across different sized datasets. SaTE shows strong performance with as few as 128 data points across the tested scenarios. The model experiences diminishing returns in performance as the number of data points is increased, with 512 data points achieving more than 99% of the performance of a model trained on 8000 randomly selected data points. This highlights how topology pruning reduces # of data points for effective training without compromising performance.

5.4 Satisfied Demand vs. Traffic Intensity

Method: We evaluate traffic allocation performance on Starlink using satisfied demand in online settings, accounting for TE computational latency. As detailed in Sec. 4, the existing traffic allocation remains active until a new one is computed. For SaTE, this occurs every second, while for other baselines, it occurs at intervals corresponding to their average computational latency shown in Fig. 8 (a), specifically 47 s, 25 s, and 54 s for Gurobi, PoP, and ECMP with WF. Experiments are conducted on Starlink with two cross-shell link scenarios: lasers and ground relays. We consider four levels of traffic intensity: 125, 250, 375, 500 flows/s, representing the average number of arriving flows per second.

Results: Fig. 10 (a & b) show the online satisfied demand versus traffic intensity. Overall, SaTE achieves the highest online satisfied demand, with an average improvement of 23.5% (via lasers) and 46.6% (via ground relays) over the respective best-performing baselines [24, 35] across various intensities, satisfying 11.0% (via lasers) and 17.7% (via ground relays) more traffic demand. We also observe that satisfied demand decreases as traffic loads increase due to the successive saturation of link capacity (200 Mbps). The distributed Satellite Routing performs the worst under heavy load, due to the lack of a holistic network view. Since Teal isn’t feasible on Starlink, we compare SaTE with Teal on 396 satellites. SaTE outperforms Teal by an average of 17.4% (via lasers) and 19.8% (via ground relays) shown in Fig. 10 (c), attributed to SaTE’s heterogeneous satellite TE graph design.



(a) Starlink, via lasers (b) Starlink, via ground relays (c) 396 satellites (d) Relative Satisfied Demand

Figure 10: SaTE's performance across various traffic intensity and scales: (a & b) In Starlink, SaTE outperforms the respective best-performing baseline [24, 35] by an average satisfied demand improvement of 23.5% (via lasers) and 46.6% (via ground relays). (c) In Mid-Size 1, where Teal can be trained, SaTE shows an improvement of 17.4% (via lasers) and 19.8% (via ground relays) over Teal [78]. (d) Separate models trained on each scale maintain robust performance, while a model trained on 396-scale shows generalization to other scales.

While SaTE optimizes total throughput, we discuss the achieved throughput for individual flows in Appendix H.4. Indeed, some flows experience partial satisfaction—a common limitation of centralized TE algorithms focused on global objectives. Mechanisms like fairness [22] or resource reservation [76] could enhance critical flow performance. Appendix H.1 evaluates offline performance without accounting for TE computation delay's impact on throughput. In offline mode, SaTE achieves near-optimal demand, slightly below Gurobi [24], but outperforms all baselines *online* with its 17 ms computation. We also evaluate the impact of link failures on throughput performance (Appendix H.3) and SaTE's applicability to different TE objectives (Appendix H.2).

5.5 Model Generalization Across Scales

Method: We previously trained separate models for each scale and tested them on unseen topologies and traffic matrices within the same scale. Now, we evaluate SaTE model's ability to generalize across scales by testing a model trained on one scale with different numbers of satellites. This can be useful for constellations undergoing gradual expansion (e.g., launches), decommissioning, unexpected node failures, or orbital maneuvers. Specifically, we train a model on 396 satellites and test it on various scales, comparing its performance with models trained on each scale using the ratio of satisfied demand relative to the optimal offline allocation [24] without considering its TE computation delay.

Results: Fig. 10 (d) shows SaTE's performance across different scales. Models trained separately on each scale maintain robust performance, achieving over 80% of the optimal offline allocation. While the 396-satellite model applied to other scales shows a slight decline—averaging 18.2%, 6.3%, and 15.9% lower for scales of 66, 1584, and 4236, respectively—it still improves satisfied demand by 3.8% (via lasers) and 23.2% (via ground relays) over the respective best-performing baselines in Starlink with 4236 satellites. Overall, a model trained on a single scale shows generalization across unseen scales, enabled by SaTE's satellite graph design, which allows the entire TE problem to be modeled and solved using GNNs alone, leveraging their inductive learning capabilities. The observed performance decline also suggests the need to retrain the model on the up-to-date constellation state, when feasible, to maintain optimal performance.

6 Related Work

Traffic Engineering and Fast Computation: With the development of software-defined networking (SDN), network operators can steer traffic and orchestrate resources from a global network view and perform online traffic control through programmable switches based on P4 [8, 39, 62]. Centralized TE solutions that use heuristic algorithms [29, 38, 71, 73] outperform conventional approaches in performance metrics, including throughput [1, 55] and link utilization [35]. As network scales increase, commercial solvers [24] require minutes, even hours to compute TE for thousands of nodes. Some TE approaches [1, 22, 55] tackles this by decomposing TE into sub-problems with reduced complexity [60]. However, these methods fail to achieve low computational latency for mega-constellations due to their polynomial increase in complexity as the network scales. While recent learning-based approaches [2, 23, 61, 78] accelerate TE for WANs via GPU parallelization, some require costly retraining when network topologies change, while others are not inherently designed for throughput maximization [2]—our primary TE objective. In contrast, SaTE rapidly computes throughput maximization for topology-changing mega-constellations.

Routing and Traffic Engineering for Satellite Networks: Recent routing protocols for satellite networks are more efficient than Dijkstra, calculating feasible routes by leveraging the grid structure of satellite networks and ISL distances [15, 45]. For constellations without direct ISLs, routing using ground relays has also been explored [27, 28]. Layered on top of routing, existing TE solutions for satellite networks manage the traffic flows on feasible routes by applying QoS constraints [4, 14, 17, 46] or congestion avoidance [16, 51, 56, 64]. However, these approaches lack centralized control and do not explicitly optimize for desirable network characteristics. [33] adopts the same graph design as [78], facing similar scalability and frequent retraining issues. In contrast, SaTE introduces a TE algorithm designed for low-latency computation and generalization to large-scale, dynamic satellite constellations.

7 Discussion and Limitations

Fine-tuning Considerations: As shown in Sec. 5.5, SaTE's model trained on a specific scale has the potential to generalize to other scales, with slight performance degradation. To further improve performance in such cases, appropriate fine-tuning using techniques like curriculum learning [25]—which incrementally trains the model with data from other scales—may be considered. This is particularly relevant for companies gradually expanding their satellite networks.

Starlink's Workflow Interval: Recent studies reveal that Starlink operates on a globally synchronized 15-second workflow interval for periodic network reconfigurations [52] However, this interval corresponds to the frequency of control-plane operations—including routing updates and policy reconfigurations—which is longer than the physical stability duration of the link topology, as observed in Fig. 4 (a). If the 15-second interval is applied to the ISL-enabled scenario considered in this paper, approximately 10–20% of ISLs become unstable or are excluded, as illustrated in Fig. 4 (b). By significantly reducing the computation bottleneck with an average runtime of 17 ms, SaTE offers the potential to shorten this interval and improve network performance.

Centralized Control in Satellite Networks: SaTE's TE computation operates within the centralized SDN for satellite networks [6, 19, 36], where satellites are managed by a centralized control center [5, 43, 83]. While the bandwidth required for control messages in such frameworks poses challenges, especially for constellations with thousands of satellites, Starlink's planned laser links (up to 200 Gbps) provide sufficient capacity for traffic rule distribution. Additionally, strategies like improved controller placement [13], multi-controller load balancing [31], and edge-assisted traffic control [10, 77] can further mitigate bandwidth overhead.

8 Conclusion

This paper presents SaTE, a low-latency TE algorithm designed for dynamic, large-scale satellite constellations. We present an analysis of how frequently topology changes in a large and dense satellite constellation with over thousands of nodes. We present a graph design that comprehensively models the satellite TE problem, and is learned solely using GNN layers, enabling fast TE computation without relying on additional DNN layers that may restrict its generalizability. SaTE then develops a dataset pruning method to make model training feasible on commercial GPUs. We present extensive data-driven simulation results of SaTE on Starlink. This work does not raise any ethical issues.

Acknowledgement

We thank the members of the AIoT Group at NUS for their insightful discussions, our shepherd, Guyue Liu, and the anonymous reviewers for their constructive feedback. This work is supported by the NUS-PYP scheme and the Singapore Ministry of Education AcRF Tier 1 Grant (251RES2519).

References

[1] Firas Abuzaid, Srikanth Kandula, Behnaz Arzani, Ishai Menache, Matei Zaharia, and Peter Bailis. 2021. Contracting wide-area network topologies to solve flow problems quickly. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. 175–200.

[2] Abd AlRhman AlQiam, Yuanjun Yao, Zhaodong Wang, Satyajeet Singh Ahuja, Ying Zhang, Sanjay G. Rao, Bruno Ribeiro, and Mohit Tawarmalani. 2024. Transferable Neural WAN TE for Changing Topologies. In *Proceedings of the ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 86–102. https://doi.org/10.1145/3651890.3672237

[3] 5G Americas. 2024. 5G Non-Terrestrial Networks. https://www.3gpp.org/news-events/partner-news/5ga-paper-ntn.

[4] Matteo Berioli, Anton Donner, Riccardo Menichelli, and Markus Werner. 2003. Mpls traffic engineering for leo satellite constellation networks. In *21st International Communications Satellite Systems Conference and Exhibit*. 2327.

[5] Lionel Bertaux, Samir Medjiah, Pascal Berthou, Slim Abdellatif, Akram Hakiri, Patrick Gelard, Fabrice Planchou, and Marc Bruyere. 2015. Software defined networking and virtualization for broadband satellite networks. *IEEE Communications Magazine* 53, 3 (2015), 54–60. https://doi.org/10.1109/MCOM.2015.7060482

[6] Lionel Bertaux, Samir Medjiah, Pascal Berthou, Slim Abdellatif, Akram Hakiri, Patrick Gelard, Fabrice Planchou, and Marc Bruyere. 2015. Software defined networking and virtualization for broadband satellite networks. *IEEE Communications Magazine* 53, 3 (2015), 54–60.

[7] Debopam Bhattacherjee, Simon Kassing, Melissa Licciardello, and Ankit Singla. 2020. In-orbit computing: An outlandish thought experiment?. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*. 197–204.

[8] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (jul 2014), 87–95. https://doi.org/10.1145/2656877.2656890

[9] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How Attentive are Graph Attention Networks? arXiv:cs.LG/2105.14491 https://arxiv.org/abs/2105.14491

[10] Li Chen, Justinas Lingys, Kai Chen, and Feng Liu. 2018. AuTO: Scaling Deep Reinforcement Learning for Datacenter-Scale Automatic Traffic Optimization. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 191–205. https://doi.org/10.1145/3230543.3230551

[11] Pulak K Chowdhury, Mohammed Atiquzzaman, and William Ivancic. 2006. Handover schemes in satellite networks: State-of-the-art and future research directions. *IEEE Communications Surveys & Tutorials* 8, 4 (2006), 2–14.

[12] Federal Communications Commission. 2024. Advancing a Framework for Supplemental Coverage from Space. https://www.fcc.gov/document/advancing-framework-supplemental-coverage-space.

[13] Tamal Das, Vignesh Sridharan, and Mohan Gurusamy. 2020. A Survey on Controller Placement in SDN. *IEEE Communications Surveys & Tutorials* 22, 1 (2020), 472–503. https://doi.org/10.1109/COMST.2019.2935453

[14] Arjan Durresi, Mimoza Durresi, and Fatos Xhafa. 2007. MPLS traffic engineering in satellite networks. In *First International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'07)*. IEEE, 19–26.

[15] E. Ekici, I.F. Akyildiz, and M.D. Bender. 2001. A distributed routing algorithm for datagram traffic in LEO satellite networks. *IEEE/ACM Transactions on Networking* 9, 2 (2001), 137–147. https://doi.org/10.1109/90.917071

[16] Eylem Ekici, Ian F Akyildiz, and Michael D Bender. 2001. A distributed routing algorithm for datagram traffic in LEO satellite networks. *IEEE/ACM Transactions on networking* 9, 2 (2001), 137–147.

[17] ETSI. 2011. *ETSI TS 102 856 V1.1.1 Part 2: Negotiation and management of MPLS labels and MPLS signalling with attached networks*. Technical Specification ETSI TS 102 856 V1.1.1. ETSI (European Telecommunications Standards Institute). https://www.etsi.org/deliver/etsi_ts/102800_102899/10285602/01.01.01_60/ts_10285602v010101p.pdf

[18] Federal Communications Commission. 2022. SPACEX NON-GEOSTATIONARY SATELLITE SYSTEM. FCC Report. https://fcc.report/IBFS/SAT-MOD-20200417-00037/2274316.pdf

[19] Ramon Ferrus, Oriol Sallent, Toufik Ahmed, and Riccardo Fedrizzi. 2017. Towards SDN/NFV-enabled satellite ground segment systems: End-to-end traffic engineering use case. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 888–893.

[20] Center for International Earth Science Information Network CIESIN Columbia University. 2018. Gridded Population of the World, Version 4 (GPWv4): Population Density, Revision 11. https://doi.org/10.7927/H49C6VHW Accessed: 2024.

[21] Ada Fort, Marco Mugnaini, Giacomo Peruzzi, and Alessandro Pozzebon. 2022. Reliability analysis of an IoT satellite facility for remote monitoring and asset tracking within marine environments. In *2022 IEEE International Workshop on Metrology for the Sea; Learning to Measure Sea Health Parameters (MetroSea)*. IEEE, 138–142.

[22] Amitabha Ghosh, Sangtae Ha, Edward Crabbe, and Jennifer Rexford. 2013. Scalable Multi-Class Traffic Management in Data Center Backbone Networks. *IEEE Journal on Selected Areas in Communications* 31, 12 (2013), 2673–2684. https://doi.org/10.1109/JSAC.2013.131208

[23] Fei Gui, Songtao Wang, Dan Li, Li Chen, Kaihui Gao, Congcong Min, and Yi Wang. 2024. RedTE: Mitigating subsecond traffic bursts with real-time and distributed traffic engineering. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 71–85.

[24] LLC Gurobi Optimization. 2022. Gurobi Optimizer Reference Manual, 2021. Online.

[25] Guy Hacohen and Daphna Weinshall. 2019. On the power of curriculum learning in training deep networks. In *International conference on machine learning*. PMLR, 2535–2544.

[26] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[27] Mark Handley. 2018. Delay is Not an Option: Low Latency Routing in Space. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks (HotNets '18)*. Association for Computing Machinery, New York, NY, USA, 85–91. https://doi.org/10.1145/3286062.3286075

[28] Mark Handley. 2019. Using ground relays for low-latency wide-area routing in megaconstellations. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*. 125–132.

[29] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. 2013. Achieving High Utilization with Software-Driven WAN. *SIGCOMM Comput. Commun. Rev.* 43, 4 (aug 2013), 15–26. https://doi.org/10.1145/2534169.2486012

[30] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, Steve Padgett, Faro Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jonathan Zolla, Joon Ong, and Amin Vahdat. 2018. B4 and after: Managing Hierarchy, Partitioning, and Asymmetry for Availability and Scale in Google's Software-Defined WAN. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 74–87. https://doi.org/10.1145/3230543.3230545

[31] Tao Hu, Zehua Guo, Peng Yi, Thar Baker, and Julong Lan. 2018. Multi-controller based software-defined networking: A survey. *IEEE access* 6 (2018), 15980–15996.

[32] Yunhui Huang, Xiaofeng Jiang, Shuangwu Chen, Feng Yang, and Jian Yang. 2022. Pheromone Incentivized Intelligent Multipath Traffic Scheduling Approach for LEO Satellite Networks. *IEEE Transactions on Wireless Communications* 21, 8 (2022), 5889–5902. https://doi.org/10.1109/TWC.2022.3144189

[33] Yunxue Huang, Dong Yang, Bohao Feng, Aleteng Tian, Ping Dong, Shui Yu, and Hongke Zhang. 2024. A GNN-Enabled Multipath Routing Algorithm for Spatial-Temporal Varying LEO Satellite Networks. *IEEE Transactions on Vehicular Technology* (2024), 1–15. https://doi.org/10.1109/TVT.2023.3333848

[34] Iridium Communications Inc. 2024. Iridium Satellite Network. https://www.iridium.com. Accessed: 2024.

[35] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. 2013. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 3–14.

[36] Weiwei Jiang. 2023. Software defined satellite networks: A survey. *Digital Communications and Networks* 9, 6 (2023), 1243–1264.

[37] Joey Roulette. 2024. SpaceX to sell satellite laser links that speed in-space communication to rivals. https://www.reuters.com/technology/space/spacex-says-plans-sell-satellite-laser-links-commercially-2024-03-19/

[38] Srikanth Kandula, Ishai Menache, Roy Schwartz, and Spandana Raj Babbula. 2014. Calendaring for wide area networks. In *ACM SIGCOMM computer communication review*, Vol. 44. ACM, 515–526.

[39] Sukhveer Kaur, Krishan Kumar, and Naveen Aggarwal. 2021. A review on P4-Programmable data planes: Architecture, research efforts, and future directions. *Computer Communications* 170 (2021), 109–129. https://doi.org/10.1016/j.comcom.2021.01.027

[40] Boris Knyazev, Graham W Taylor, and Mohamed Amer. 2019. Understanding attention and generalization in graph neural networks. *Advances in neural information processing systems* 32 (2019).

[41] Sastri Kota and Giovanni Giambene. 2019. Satellite 5G: IoT use case for rural areas applications. In *Proceedings of the Eleventh International Conference on Advances in Satellite and Space Communications-SPACOMM*. 24–28.

[42] Alex Kulesza, Ben Taskar, et al. 2012. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning* 5, 2–3 (2012), 123–286.

[43] Taixin Li, Huachun Zhou, Hongbin Luo, and Shui Yu. 2018. SERvICE: A Software Defined Framework for Integrated Space-Terrestrial Satellite Communication. *IEEE Transactions on Mobile Computing* 17, 3 (2018), 703–716. https://doi.org/10.1109/TMC.2017.2732343

[44] Yuanjie Li, Hewu Li, Wei Liu, Lixin Liu, Wei Zhao, Yimei Chen, Jianping Wu, Qian Wu, Jun Liu, Zeqi Lai, et al. 2023. A networking perspective on starlink's self-driving leo mega-constellation. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–16.

[45] Yuanjie Li, Lixin Liu, Hewu Li, Wei Liu, Yimei Chen, Wei Zhao, Jianping Wu, Qian Wu, Jun Liu, and Zeqi Lai. 2024. Stable Hierarchical Routing for Operational LEO Networks. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 296–311.

[46] Fei Long. 2014. *Satellite network robust QoS-aware routing*. Springer.

[47] Sami Ma, Yi Ching Chou, Haoyuan Zhao, Long Chen, Xiaoqiang Ma, and Jiangchuan Liu. 2023. Network characteristics of LEO satellite constellations: A Starlink-based measurement from end users. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[48] Kris Maine, Carrie Devieux, and Pete Swan. 1995. Overview of IRIDIUM satellite network. In *Proceedings of WESCON'95*. IEEE, 483.

[49] Starlink Map. 2024. Satellitemap. https://satellitemap.space/. Accessed: 2024.

[50] Jonathan C McDowell. 2020. The low earth orbit satellite population and impacts of the SpaceX Starlink constellation. *The Astrophysical Journal Letters* 892, 2 (2020), L36.

[51] Scott Moeller, Avinash Sridharan, Bhaskar Krishnamachari, and Omprakash Gnawali. 2010. Routing without routes: The backpressure collection protocol. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. 279–290.

[52] Nitinder Mohan, Andrew E. Ferguson, Hendrik Cech, Rohan Bose, Prakita Rayyan Renatin, Mahesh K. Marina, and Jörg Ott. 2024. A Multifaceted Look at Starlink Performance. In *Proceedings of the ACM Web Conference 2024 (WWW '24)*. Association for Computing Machinery, New York, NY, USA, 2723–2734. https://doi.org/10.1145/3589334.3645328

[53] Matthias Motzigemba, Herwig Zech, and Philipp Biller. 2019. Optical Inter Satellite Links for Broadband Networks. In *2019 9th International Conference on Recent Advances in Space Technologies (RAST)*. 509–512. https://doi.org/10.1109/RAST.2019.8767795

[54] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning Distributed Representations of Graphs. *arXiv preprint arXiv:1707.05005* (2017).

[55] Deepak Narayanan, Fiodar Kazhamiaka, Firas Abuzaid, Peter Kraft, Akshay Agrawal, Srikanth Kandula, Stephen Boyd, and Matei Zaharia. 2021. Solving Large-Scale Granular Resource Allocation Problems Efficiently with POP. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP '21)*. Association for Computing Machinery, New York, NY, USA, 521–537. https://doi.org/10.1145/3477132.3483588

[56] Michael James Neely. 2003. *Dynamic power allocation and routing for satellite and wireless networks with time varying channels*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[57] Mark Newman. 2010. *Networks: An Introduction*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199206650.001.0001

[58] John Nguyen. 2020. Overview of existing and future advanced satellite systems. In *Satellite Systems-Design, Modeling, Simulation and Analysis*. IntechOpen.

[59] NVIDIA. 2024. NVIDIA A100 Tensor Core GPU Architecture. https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf. Accessed: 2024.

[60] D.P. Palomar and Mung Chiang. 2006. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications* 24, 8 (2006), 1439–1451. https://doi.org/10.1109/JSAC.2006.879350

[61] Yarin Perry, Felipe Vieira Frujeri, Chaim Hoch, Srikanth Kandula, Ishai Menache, Michael Schapira, and Aviv Tamar. 2023. DOTE: Rethinking (Predictive) WAN Traffic Engineering. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 1557–1581. https://www.usenix.org/conference/nsdi23/presentation/perry

[62] Debobroto Das Robin and Javed I. Khan. 2022. P4TE: PISA switch based traffic engineering in fat-tree data center networks. *Computer Networks* 215 (2022), 109210. https://doi.org/10.1016/j.comnet.2022.109210

[63] Juan Luis Cano Rodríguez and Jorge Martínez Garrido. 2022. poliastro: a Python library for interactive astrodynamics.. In *SciPy*. 136–146.

[64] Jung Ryu, Lei Ying, and Sanjay Shakkottai. 2010. Back-pressure routing for intermittently connected networks. In *2010 Proceedings IEEE INFOCOM*. IEEE, 1–5.

[65] Sandvine. 2024. 2024 Global Internet Phenomena Report. https://www.sandvine.com/phenomena.

[66] Space Exploration Technologies Corp. (SpaceX). 2024. Starlink. https://www.starlink.com/

[67] SpaceX. 2024. 10,000 operational space lasers for the constellation. https://x.com/SpaceX/status/1764818522813931993.

[68] Starlink. 2024. Starlink Technology. https://www.starlink.com/technology.

[69] Statista. 2024. Number of internet users by country 2023. https://www.statista.com/statistics/262966/number-of-internet-users-in-selected-countries/. Accessed: 2024.

[70] PyG Team. 2024. Heterogeneous Graph Learning. https://pytorch-geometric.readthedocs.io/en/latest/notes/heterogeneous.html#heterogeneous-graph-learning. Accessed: 2024.

[71] Slavica Tomovic and Igor Radusinovic. 2019. Toward a Scalable, Robust, and QoS-Aware Virtual-Link Provisioning in SDN-Based ISP Networks. *IEEE Transactions on Network and Service Management* 16, 3 (2019), 1032–1045. https://doi.org/10.1109/TNSM.2019.2929161

[72] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[73] Jason Min Wang, Ying Wang, Xiangming Dai, and Brahim Bensaou. 2019. SDN-Based Multi-Class QoS Guarantee in Inter-Data Center Communications. *IEEE Transactions on Cloud Computing* 7, 1 (2019), 116–128. https://doi.org/10.1109/TCC.2015.2491930

[74] Zhaokang Wang, Yunpan Wang, Chunfeng Yuan, Rong Gu, and Yihua Huang. 2021. Empirical analysis of performance bottlenecks in graph neural network training and inference with GPUs. *Neurocomputing* 446 (2021), 165–191.

[75] Xitao Wen, Bo Yang, Yan Chen, Li Erran Li, Kai Bu, Peng Zheng, Yang Yang, and Chengchen Hu. 2016. RuleTris: Minimizing Rule Update Latency for TCAM-Based SDN Switches. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. 179–188. https://doi.org/10.1109/ICDCS.2016.41

[76] Brian Wolshon and Anurag Pande. 2016. *Traffic engineering handbook.* John Wiley & Sons.

[77] Hao Wu, Jian Yan, and Linling Kuang. 2024. Asynchronous Multi-Class Traffic Management in Wide Area Networks. *IEEE Transactions on Network and Service Management* (2024), 1–1. https://doi.org/10.1109/TNSM.2024.3354793

[78] Zhiying Xu, Francis Y. Yan, Rachee Singh, Justin T. Chiu, Alexander M. Rush, and Minlan Yu. 2023. Teal: Learning-Accelerated Optimization of WAN Traffic Engineering. In *Proceedings of the ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 378–393. https://doi.org/10.1145/3603269.3604857

[79] Dongxu Yang, Junhong Liu, Jiaxing Qi, and Junjie Lai. 2022. WholeGraph: A Fast Graph Neural Network Training Framework with Multi-GPU Distributed Shared Memory Architecture. In *Proceedings of the 35th International Conference for High Performance Computing, Networking, Storage and Analysis*. 54:1–54:14.

[80] Jin Y Yen. 1971. Finding the k shortest loopless paths in a network. *management Science* 17, 11 (1971), 712–716.

[81] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2021. Dataset Condensation with Gradient Matching. In *International Conference on Learning Representations*. https://openreview.net/forum?id=mSAKhLYLSsl

[82] Da Zheng, Xiang Song, Chengru Yang, Dominique LaSalle, and George Karypis. 2022. Distributed Hybrid CPU and GPU training for Graph Neural Networks on Billion-Scale Heterogeneous Graphs. In *Proceedings of the 28th ACM Conference on Knowledge Discovery and Data Mining*. 4582–4591.

[83] Gao Zheng, Ning Wang, and Regius Rahim Tafazolli. 2024. SDN in Space: A Virtual Data-Plane Addressing Scheme for Supporting LEO Satellite and Terrestrial Networks Integration. *IEEE/ACM Transactions on Networking* 32, 2 (2024), 1781–1796. https://doi.org/10.1109/TNET.2023.3330672

Appendices

Appendices are supporting material that has not been peer-reviewed.

A Appendix: TE Model in Satellite Networks

Based on the notations listed in Table 3, the optimization problem of TE can be expressed as

$$\begin{aligned}
&\text{maximize } \mathcal{U} = \sum_{f\in\mathcal{F}_t}\sum_{p\in\mathcal{P}_f} x_{fp} && (2.a)\\
&\text{subject to } \sum_{f\in\mathcal{F}_t}\sum_{p\in\mathcal{P}_f} \Phi_{pe}x_{fp} \le C_e, \forall e && (2.b)\\
&\sum_{f:f\uparrow n}\sum_{p\in\mathcal{P}_f} x_{fp} \le C_n^{up}, \forall n && (2.c)\\
&\sum_{f:f\downarrow n}\sum_{p\in\mathcal{P}_f} x_{fp} \le C_n^{dn}, \forall n && (2.d)\\
&\sum_{p\in\mathcal{P}_f} x_{fp} \le d_f, \forall f && (2.e)\\
&\text{variables } x_{fp} \ge 0, \forall f, p && (2.f)
\end{aligned} \quad (2)$$

where $\mathcal{U}$ represents network throughput, $\mathcal{F}_t$ includes all ongoing flows at time t and $\mathcal{P}_f$ represents all feasible paths for flow f, where a path $p \in \mathcal{P}_f$ is a sequence of links that connects the source and destination nodes of flow f. The objective function (2.a) can be adapted to accommodate different TE strategies. Two common examples include:

$$\begin{aligned}
&\text{Min-Max Link Utilization: } \min\max_e \sum_{f\in\mathcal{F}_t}\sum_{p\in\mathcal{P}_f} \frac{\Phi_{pe}x_{fp}}{C_e},\\
&\text{Maximize Network Utility: } \max \sum_{f\in\mathcal{F}_t} u_f\Big(\sum_{p\in\mathcal{P}_f} x_{fp}\Big),
\end{aligned} \quad (3)$$

depending on the network operator's objectives. Constraint (2.b) ensures that the allocated traffic on each ISL does not exceed its capacity. Constraint (2.c) limits the total data rate to satellite n to its uplink capacity, where $f \uparrow n$ indicates that flow f accesses satellite n via its uplink. Constraint (2.d) similarly applies to the downlink. Constraint (2.e) ensures that the allocated bandwidth to a flow does not exceed its demand, preventing resource over-provisioning and waste. In satellite networks, the uplink and downlink capacities depend on several factors, including the multiple access mechanism, antenna settings, and distance, implying that constraints (2.c & d) can be extended to more complex physical layer formulas. It is important to note that SaTE focuses solely on network layer optimization.

The network performance largely depends on the choice of objective function. There is a long-standing trade-off between efficiency (e.g., total throughput) and fairness in TE. Some operators aim to maximize throughput, while others prioritize fairness by employing log-based objectives. The concavity of the log function inherently limits any single flow from monopolizing resources, offering a soft fairness guarantee.

Symbol	Description
$f, \mathcal{F}_t$	Flow and flow set at TE slot t
$p, \mathcal{P}_f$	Path and candidate path set of f
x_{fp}	Bandwidth allocated to flow f on path p
Φ_{pe}	Relation between path p and link e. If path p contains e, $\Phi_{pe} = 1$; otherwise, $\Phi_{pe} = 0$
C_e	ISL capacity
C_n^{up}	Up-link capacity of satellite n
C_n^{dn}	Down-link capacity of satellite n

Table 3: List of Key Notations

The traffic and path pruning technique described in Sec. 3.4 does not affect the correctness of the TE formulation. Specifically, the pruning removes only trivial traffic entries, i.e., source-destination pairs with zero demand. For such pairs, the allocated flow must be 0 by definition—it contributes neither to bandwidth usage nor to the optimization objective (e.g., total throughput). Mathematically, removing such variables and their associated constraints—including the corresponding candidate paths—does not change the feasible region, the optimal solution, or the optimal objective value of the TE problem. Therefore, passing the pruned traffic matrix and path set to the GNN leads to solving a smaller but equivalent TE problem.

B Appendix: Training the GNN Model

We construct a blend of supervised learning and penalized optimization to guide the model toward optimal traffic allocation. By balancing the ratio factors λ_{flow} and λ_{balance}, SaTE is able to solve the constrained TE optimization problem with the guidance of a reference value (i.e., the loss function in supervised learning, $\mathcal{L}_{\text{supervised}}$). The mixed loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{supervised}} + \frac{-\lambda_{\text{flow}} \cdot \text{total\_flow} + \sum_i \alpha_i \cdot \text{over\_flow}_i}{\lambda_{\text{balance}} \cdot \lambda_{\text{flow}} \cdot \text{total\_demand}}, \tag{4}$$

where

$$\alpha_i = \exp\left(\min\left(\frac{\text{utilization}_i}{\text{capacity}_i}, \alpha_{\max}\right)\right). \tag{5}$$

In this mixed loss function, total_flow represents the sum of allocated traffic bounded by constraints, and over_flow$_i$ represents the overloaded traffic that exceeds the capacity of each link.

Our approach of combining supervised learning and penalized optimization enables SaTE to learn from the reference value while being guided by the penalty term for constraint violations. To mitigate the common issues associated with deeper GNNs—such as over-smoothing and vanishing gradients—we incorporate residual connections into our GNN modules.

To optimize the associated hyperparameters, we determined λ_{flow}, λ_{balance}, and $\alpha_{\max}$ through grid search. For selecting the number of message-passing layers in each module, we determine it based on inference time rather than validation performance. Specifically, we use the minimum number of layers possible without incurring performance degradation. The rationales are that: 1) Our main focus is low inference latency which does not discriminate between training or testing dataset, and more importantly 2) As satellite datasets share similar topological structures (as we have mentioned in our topology pruning section), we performed only minimal hyperparameter tuning to avoid creating test-like feedback loop.

C Appendix: Shortest Paths in Multi-Shell Satellite Constellations.

Training GNNs in mega-constellations like Starlink is time-consuming using traditional k-shortest path algorithms (e.g., Yen's algorithm[80]) due to their polynomial complexity relative to node scales. On the Starlink topology with 4236 satellites, pre-computing 10 shortest paths for each source-destination pair can take hundreds of seconds, leading to days of computation for just a thousand pairs. To enable fast computing, we design a rapid shortest path algorithm tailored to the multi-shell grid structure of mega-constellations like Starlink.

Intra-Shell Paths: In the grid topology of each shell, we label each satellite by its (*orbit number, intra-orbit satellite number*) coordinate (see Fig. 11 (a)). For two satellites with coordinates (x_1, y_1) and (x_2, y_2), the minimum number of hops is the Manhattan distance $|x_1 - x_2| + |y_1 - y_2|$. Up to $k = \binom{|x_1-x_2|+|y_1-y_2|}{|x_1-x_2|}$ paths with the minimum hops can then be computed. For example, in Fig. 11 (a), the three possible paths in Shell 1 from (4,2) to (2,1) are (4,2) → (3,2) → (2,2) → (2,1), (4,2) → (3,2) → (3,1) → (2,1) and (4,2) → (4,1) → (3,1) → (2,1).

Inter-Shell Paths: Finding candidate paths in multi-shell scenarios involves three steps. In Fig. 11 (a), assume the source is (2,2) in Shell 2 and the destination is (2,1) in Shell 1. First, a recursive algorithm finds the nearest satellite to the source among all nodes in Shell 2 that have a cross-shell link to an intermediate node in Shell 1, returning (3,2). Second, we compute up to k paths with the minimum hops from the intermediate node (i.e., (4,2)) to the destination. Third, we concatenate one shortest intra-shell path from the source to (3,2) in Shell 1 and the k shortest intra-shell paths from

(4,2) to the destination in Shell 2, minimizing hops on higher shells with sparser links to avoid congestion.

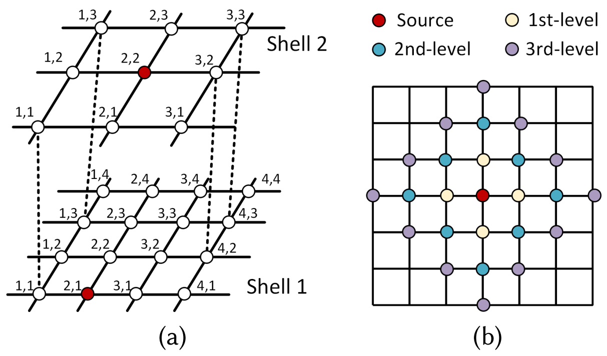


Figure 11: (a) An example of multi-shell satellite grids. (b) An example of the recursion algorithm.

The recursion algorithm successively traverses the satellites with an m-hop distance from the source during the m^{th}-level recursion, as shown in Fig. 11 (b). Since cross-shell links via lasers are dense, the algorithm typically returns a result after traversing a few levels. However, when using cross-shell links via ground relays, the algorithm is less efficient at finding the nearest satellite connected to a ground station due to the limited locations of ground relays (primarily on land). In this case, we directly calculate the distances to satellites connected to ground stations and select satellite α with the minimum hops to the source. The complete path is then obtained by concatenating the path from the source to α, the ground station β connected to α, the satellite γ in the destination shell that connects to β, and k shortest paths from β to the destination.

When the destination is in a higher shell, the algorithm works similarly: we find the nearest satellite α to the destination with a cross-shell link (or links) to an intermediate node β in the source shell. We then determine intra-shell paths from the source to β and concatenate them with the path from α to the destination.

Impact on Communication Delay: The communication latency experienced by users primarily depends on path delays, which are influenced by routing and access strategies, as illustrated in Fig. 12. We endeavor to have both communication parties access the same layer of satellites to promote stabler path delays. An in-depth discussion on routing and access strategies is beyond the scope of this paper.

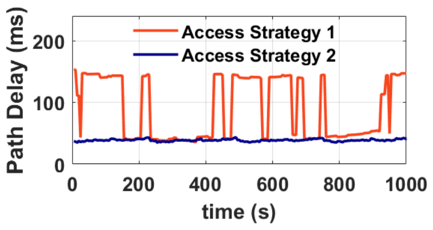


Figure 12: For two typical users at Frankfurt and Singapore, the path delay differs across two access strategies: 1. Users can access any visible satellites; 2. End users of a connection access satellites in the same orbital shell.

D Appendix: Supplementary Details in Satellite TE

In the TE workflow described in Sec. 2.2, the control center acquires the traffic matrix and distributes the computed traffic rules to satellite nodes.

Leveraging the traffic information collected during the connection establishment procedure, the control center can estimate bandwidth demands as follows:

- For persistent flows (e.g., VoIP, video streaming, file transfers): The control center can infer bandwidth requirements from communication types and standards. For example, VoIP using the ITU-T G.711 standard typically requires 64 kbps. The control center can then aggregate such demands between each satellite pair.
- For background flows: The average bandwidth needed before a specified deadline (if any) can serve as the demand indicator.
- For bursty flows (e.g., image transmission during a chat): These can preempt bandwidth from background flows. Given their typically small volume, their demand can be implicitly accounted for without explicit prediction.

Unlike traffic matrix acquisition, traffic rule distribution requires message propagation between the control center and satellites, and its duration is determined by the network's maximum round-trip time. For Starlink, assuming the control center is located in Houston, the time required for traffic demand acquisition and rule distribution is calculated based on the geographical positions of satellites and the control center. The control center interacts with satellites via ISLs and direct satellite links. Propagation delays are computed by dividing the distance of each ISL or direct link to the control center by the speed of light. The total end-to-end delay for a satellite is the sum of delays along the shortest path between the control center and the satellite.

The distribution of traffic rules incurs this propagation delay. Messages from satellites directly connected to the control center are sent directly, while those from remote satellites are relayed through multiple ISL hops. For Starlink's

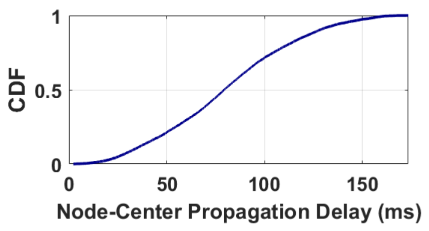


Figure 13: It takes less than 174 ms to distribute traffic rules to 4236 Starlink satellites.

4,236-satellite constellation, the propagation delays range from 2.3 ms to a maximum of 174 ms (see Fig. 13). This results in a sub-second timescale for both traffic matrix collection and rule distribution, which is significantly shorter than traditional TE computation times that can take minutes or even hours.

Unlike traffic matrix acquisition—which leverages the inherent access control mechanism to obtain demand information—traffic rule distribution incurs additional traffic overhead. Suppose there are m concurrently active node pairs, each associated with k candidate paths of average length E_l. Then, the total number of rules is approximately mkE_l. Given the sparse distribution of satellite users, m increases slowly with network scale. Moreover, since E_l usually scales as $O(\ln(n))$ due to the logarithmic diameter property of random graphs [57], the total overhead scales as $O(mk\ln(n))$, which remains negligible compared to the total link bandwidth of $O(n)$ (typically for $O(n)$ ISLs in a constellation of n satellites).

E Appendix: Dataset Pruning Based on Topology Similarity

Since the dataset size for traffic engineering can reach into the millions, a procedure is needed to prune the dataset to allow for efficient training. By pruning based on topology similarity, we enable learning from a diverse, but representative dataset. The dataset is pruned in the following manner:

- *Graph to vectorized representation.* To enable further processing, a convenient representation for each topology needs to be created. We utilize Graph2Vec [54] to generate a fixed size vector representation for each graph. This vectorized representation summarizes the graph substructures, which collectively form the graph. Two different graphs with similar topologies, and consequently similar graph substructures, will have a similar vectorized representation.
- *Determinantal Point Process sampling*. To generate a representative and diverse dataset based on the vectorized representation, we use Determinantal Point Process (DPP) to sample graphs from the vectorized representations. By maximizing the determinant of the kernel matrix derived from these vectors, DPPs ensure that the sampled subsets include a broad range of graph topologies. Maximizing the determinant entails searching for a set of maximally linearly independent vectors, which corresponds to searching for diverse network topologies. This approach guarantees that the sampled subset represents a comprehensive and varied portion of the dataset, reflecting its overall diversity.

Given the dataset to be pruned formed by n topology graphs $\{\mathcal{G}_1, \dots, \mathcal{G}_n\}$ where each $\mathcal{G}_i$ represents a topology, $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ the following procedure is followed:

- Using Graph2Vec, the dataset of topology graphs is transformed into a vector representation using $f : \mathcal{G} = (\mathcal{V}, \mathcal{E}) \to \mathbb{R}^d$ for a fixed dimensionality d. In our work, we use $d = 128$.
- Using Determinantal Point Process, the dataset is reduced to a manageable size by sampling from the transformed dataset, $\{f(\mathcal{G}_1), \dots, f(\mathcal{G}_n)\}$ to a fixed sized dataset of $\{f(\mathcal{G}_{s_1}), \dots, f(\mathcal{G}_{s_k})\}$ where k is an appropriately picked value. We pick a sufficiently large value for k such that the number of chosen topologies gives a well-trained model.
- Finally, the graphs corresponding to sampled vectorized representations form the topologies for the training dataset: $\{\mathcal{G}_{s_1}, \dots, \mathcal{G}_{s_k}\}$.

F Appendix: Message Passing Mechanism

The detailed update formula of a general embedding denoted by v_i is shown in (6).

$$\begin{aligned} v_i' &\leftarrow \Theta_s \cdot v_i + \Big\|_{k=1}^{K} \left(\sum_{j \in r(i)} \alpha_{j,i}^k (\Theta_n^k \cdot v_j + \Theta_e^k \cdot e_{j,i}) \right), \\ v_i' &\leftarrow \text{LeakyReLU}(v_i'), \end{aligned} \tag{6}$$

where Θ represents the learnable weight matrices for transforming features of the node ($s = self$), neighboring nodes ($n = neighbor$), and edge features ($e = edge$). K attention heads are utilized, and $\|$ denotes the concatenation operator. $\alpha_{j,i}^k$ denotes learnable attention coefficients, which are utilized to weigh the influence of neighboring nodes. They are calculated as:

$$\begin{aligned} \alpha_{j,i}^k = \ & \text{softmax}_i(\text{LeakyReLU}(\\ & a^T \left[\Theta_n^k \cdot v_i \parallel \Theta_n^k \cdot v_j \parallel \Theta_e^k \cdot e_{j,i} \right])), \end{aligned} \tag{7}$$

with a corresponding to a learnable vector.

G Appendix: Satellite Orbits, User Locations, and Traffic Matrix

The satellite trajectory emulation of Starlink in this paper replicates the largest actively operating satellite constellation

		Altitude	Inclination	Orbital planes	Satellites per orbit
Iridium		781 km	86.4°	6	11
Starlink	Shell 1	540 km	53.2°	72	22
	Shell 2	550 km	53.0°	72	22
	Shell 3	560 km	97.6°	6	58
	Shell 4	570 km	70°	36	20

Table 4: Orbital Parameters for Starlink and Iridium

as of April 2024: Starlink (Phase 1), with 4,236 satellites. The orbit parameters are listed in Table 4.

To evaluate SaTE's performance on various network scales, we also emulate trajectories of more satellite constellations: *(1) Iridium:* A constellation of 66 satellites in a single shell at 781 km altitude, using Ka-band ISLs [34]. *(2) Mid-Sized Constellations:* By retaining only the orbital shell 1 and 2 of Starlink and reducing the number of orbital planes by factors of 8 and 2, we created two mid-sized constellations with 396 and 1584 satellites, serving as intermediaries between Iridium and full-scale Starlink.

We generate user and gateway locations based on the uneven global distribution of population. The Earth's surface is divided into 360×180 grids, each spanning 1° in latitude and longitude. The proportion of users in each grid, denoted as α, is calculated by:

$$p_\alpha = \frac{\text{Population Density in Grid } \alpha + \gamma}{\sum_\alpha(\text{Population Density in Grid } \alpha + \gamma)}, \tag{8}$$

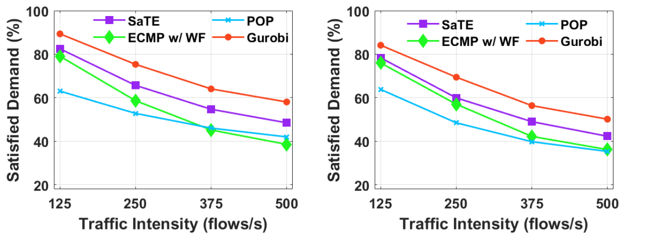
where a smoothing factor γ to enhance user representation in sparsely populated areas. Using this distribution, the traffic generator separately places 3 million Starlink users and 1,000 gateways across grids according to p_α. Users and gateways are generated in separate runs, following the same population-based distribution. Traffic flows are randomly generated for user-to-user and user-to-gateway communication, following a Poisson arrival process with a total traffic intensity of λ. The traffic intensity between grids α and β is $\lambda p_\alpha p_\beta$, aligned with user distribution. Traffic intensity is adjusted to evaluate TE performance under varying traffic loads.

H Additional Performance Evaluation

H.1 Offline Performance of Baselines

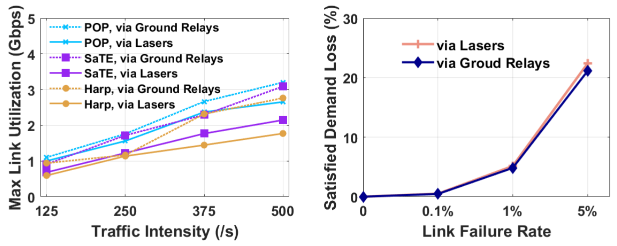
In the offline setting, we eliminate the impact of computational latency to focus solely on the quality of the traffic allocation. This idealized scenario assumes that SaTE and the baselines compute traffic allocation instantaneously.

In addition, SaTE records the second-highest offline satisfied demand, 12.8% (with laser) and 12.3% (with ground relays) lower than the theoretical upper bound [24]. It's important to note that while SaTE delivers near-optimal performance in offline mode, it outperforms all other baselines in



(a) Offline, via lasers (b) Offline, via ground relays

Figure 14: SaTE's offline performance on satisfied demand across various traffic intensity for Starlink.



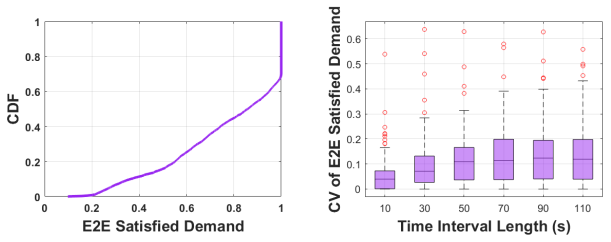
(a) Min-Max Link Utilization (b) Impact of link failures

Figure 15: (a) SaTE outperforms the baseline [55] in minimizing MLU by 21% using lasers and 13% using ground relays. (b) 1% link failures lead to less than 5.2% loss.

online scenarios due to its ultra-low-latency of about 17 ms. The performance gap between SaTE and the upper bound achieved by Gurobi arises from multiple factors, including post-correction adjustments and model degradation on unseen inputs. Bridging this gap represents a central goal for the future development of our framework.

H.2 Max Link Utilization

So far, we've trained SaTE to maximize throughput. We further evaluate SaTE's applicability in minimizing *Max Link Utilization* (MLU), which represents the bandwidth required on ISLs to accommodate all traffic demands, with lower MLU values being preferred by network operators. We train SaTE to minimize MLU in Starlink and evaluate its performance across various traffic intensities. SaTE outperforms the baseline [55] by 24.5% (via lasers) and 9.3% (via ground relays), as shown in Fig. 15 (a). Additionally, SaTE exhibits an MLU gap of 16% (laser links) and 13% (ground relays) compared to the baseline [2], which employs a targeted MLU-optimized design. This gap arises because we directly repurpose our throughput-maximizing GNN's objective function for MLU minimization, potentially retaining redundant components



(a) CDF of flow-level satisfied demand (b) CV of flow-level satisfied demand

Figure 16: (a) Some satellite pairs fulfill their input traffic demands, while others need additional mechanisms to grantee high-priority traffic. (b) Users experience a relatively stable service in a with a median CV of less than 0.12.

not perfectly suited for the simpler, less-constrained MLU optimization problem.

H.3 Impact of Satellite Link Failures

Method: We evaluate the performance of SaTE under sudden satellite link failures, which may result from hardware errors or trajectory deviation. We randomly induce link failures in Starlink at rates of 0.1%, 1%, and 5% per TE interval, making 1.2%, 8.7%, and 18.6% configured paths invalid. SaTE then allocates traffic to the remaining paths.

Results: Our model is capable of handling link failures; however, as expected, link (or node) failures can degrade overall throughput. Fig. 15 (b) shows the average loss (%) in satisfied demand induced by link failures. We observe that the loss remains below 5.2% without any rerouting when the link failure rate is under 1%. The results indicate that SaTE is adaptive to link failures, even those not encountered during training. We further note that the link failure rate of 5% corresponds to 250 concurrent laser-link failures or 115 concurrent ground-relay based link failures within 1 second across the network, which is fairly rare in practice.

H.4 Flow-level Insights

Flow-level Insights: While SaTE optimizes global throughput, we discuss the performance of individual flows between satellite pairs to reflect individual user performance. This analysis focuses on the ratio of allocated traffic to traffic demand for individual node pairs at a traffic intensity of 125 flows/s. Fig.16 (a) presents the CDF of flow-level satisfied demand across all satellite pairs with non-zero demand. Over 30% of satellite pairs fully meet their traffic demands, while others achieve partial satisfaction—a common limitation of centralized TE algorithms prioritizing global objectives. Mechanisms like fairness [22] or resource reservation [76] could improve performance for critical flows. Fig. 16 (b) shows the coefficient of variation (CV) of flow-level satisfied demand over various time spans, showing stable performance with slight CV increases over longer spans due to network dynamics and traffic shifts.