Throughput-Efficient Design and Machine Learning for Wireless Mesh Network Optimization

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Yung Fu Chen, B.S., M.S.

Graduate Program in Department of Computer Science and Engineering

The Ohio State University

2024

Dissertation Committee:

Anish Arora, Advisor

Ness Shroff

Kannan Athreya

Shaileshh Bojja Venkatakrishnan

Abstract

Wireless meshes offer a resilient and cost-effective framework where multi-hop communication occurs among mesh clients, routers, and gateways. In this framework, computing and communication are essential to guarantee capacity and achieve high performance in terms of network planning, frequency scheduling, and packet routing. In order to optimize the achievable capacity to provide high throughput-latency communication with scalability, adaptivity, and reliability against various network configurations and dynamics, my thesis studies a cross-layer approach, from network infrastructure development to communication schedule and routing path selection, to show that the problem complexity can be managed by decomposition into subproblems via approximation, even learning based solutions. My study in this thesis consists of several approaches: first, we focus on the middle-mile network optimization problem to provide broadband connectivity in rural regions with a theoretical upper bound of infrastructure cost. Second, we investigate channel hopping to achieve high spectrum utilization, interference avoidance, and jamming tolerance. Third, we look into capacity-aware routing using bounded exploration regions to high throughput and reliability with low overhead. Finally, by developing machine learning algorithms using domain knowledge of bounded exploration on the network routing problem, we study the generalizability of the learned routing policies to all uniform random graphs.

The middle-mile network optimization is to connect the last-mile networks to the core network service providers with minimal infrastructure cost and throughput constraints. It includes topology construction, tower height assignment, antenna and orientation selection, as well as transmit power assignment, which is known to be a computationally hard problem. We propose the first polynomial time approximation solution for a generalized version of the middle-mile network optimization problem, wherein point-to-point links are deployed to bridge last-mile networks.

In wireless meshes with potentially malicious external or internal interference, the throughput efficiency can be significantly improved by equipping routers with multi-radio access over multiple channels. However, multi-radio scheduling over a limited set of channels to minimize the effect of interference and maximize network performance in the presence of concurrent network flows remains a challenging problem. To the end, we propose an adaptive channel hopping algorithm for multi-radio communication, QF-MAC, that assigns per-node, per-flow local channel hopping sequences, using only one-hop neighborhood coordination. QF-MAC achieves not only a substantial enhancement of throughput and latency with low control overhead, but also robustness against network dynamics, i.e., mobility and external interference, and selective jamming attacker where a global channel hopping sequence (e.g., TSCH) fails to sustain the communication performance.

With respect to routing in wireless meshes, we first show that the length of shortest paths in networks with uniform random node distribution can be bounded with high probability. Thus, with high probability, the shortest path may be found by limiting exploration to an elliptic region whose size is a function of the network density and the Euclidean distance between the two endpoints. Then we propose a geographic

routing protocol that achieves high reliability and throughput-latency performance by forwarding packets within an ellipse whose size is bounded in a similar manner as well as by an estimate of the available capacity. The protocol, QF-Geo, selects forwarding relays within the elliptic region, prioritizing those with sufficient capacity to avoid bottlenecks caused by internal and external interference.

In order to enhance the generalizability of the routing policy to various wireless meshes, we further develop offline supervised learning and reinforcement learning with and without the prior knowledge of bounded exploration for the approximations of routing protocols. A single-copy routing policy is learned from a small set of networks and can scale to larger networks. The analysis of the performance of predicted routes shows that, without domain knowledge, the learned model at most converges to greedy forwarding; by using the knowledge of routing in a bounded region, the learned model generalizes to graphs with different levels of network size and density and outperforms greedy forwarding.

This is dedicated to my mother Mei-Yue Yu and my family.

Acknowledgments

I would not have been able to complete my Ph.D. study without the help and guidance of many good people in the past few years.

I would like to first thank my advisor, Professor Anish Arora, for his guidance and support of my research and dissertation. His insightful advice and positive attitude have deeply impacted me in overcoming many difficulties and challenges in my research and remaining optimistic about enjoying life these past few years. He always leads me to forge a solid basis for academic research engagement and explore new ideas to solve problems. I greatly appreciate his patience in guiding me to stay on the right track to conduct the research projects and complete my research topics. I have benefited a lot from his thoughtful advice, and what I have learned from him will continue to benefit my life in the future.

I would like to thank Prof. Ness B. Shroff, Prof. Kannan Athreya, and Prof.Shaileshh Bojja Venkatakrishnan for serving on my candidacy and dissertation committee. They offered helpful advice, which motivated my critical thinking and further improved my research. I would also like to thank Prof. Feng Qin and Prof. Wei-Lun Chao for being my minor coursework advisors. I strengthened my knowledge of the domains of software systems and artificial intelligence from their teaching courses. This expertise enables me to integrate my networking knowledge with research topics.

I would like to thank my TA supervisor of CSE 1110, Michelle Mallon, for all your mentoring and support, which made me a successful grader in assisting undergraduates and a helpful instructor in teaching fundamental knowledge of computing technology.

I would like to thank the former and current colleagues in my research group for their friendship and assistance over the past few years: Dr. Dhrubojyoti Roy, Dr. Sangeeta Srivastava, and Jihoon Yun. All their support helped me overcome various difficulties in coursework and research projects. They are also very enthusiastic about sharing their research and life experiences, which eased my stress about studying abroad in the first few years.

I would like to thank my father, Wei-Jau Chen, and my sister, Pei-Hsuan Chen, for their support and care during these years. Finally, I would sincerely like to thank my mother, Mei-Yue Yu, for all your unconditional love. She granted me the courage to push against many difficulties and challenges. Without her meticulous parenting, I would not have been able to achieve my current accomplishments.

Vita

October 15, 1985	Born - New Taipei City, Taiwan
2008	B.S. Computer Science
2010	M.S. Computer Science
2016-present	Graduate Teaching & Research Associate, Department of Computer Science & Engineering, The Ohio State University.

Publications

Research Publications

Y.C. Wang, Y.F. Chen, and Y.C. Tseng "Using rotatable and directional (R&D) sensors to achieve temporal coverage of objects and its surveillance application". *IEEE Transactions on Mobile Computing*, Aug. 2012.

Y.F. Chen, and A. Arora "Middle-mile network optimization in rural wireless meshes". *In IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, Aug. 2020.

T.U. Islam, J.O. Boateng, G. Zu, M. Shahid, M. Nadim, W. Xu, T. Zhang, S. Reddy, X. Li, A. Atalar, Y.F. Chen, S. Babu, H. Zhang, D. Qiao, M. Zheng, Y. Guan, O. Boyraz, A. Arora, M. Selim, M.B. Cohen "ARA PAWR: Wireless Living Lab for Smart and Connected Rural Communities". *In Proceedings of the 29th ACM Annual International Conference on Mobile Computing and Networking Demos Session (MobiCom)*, Oct. 2023 (demo).

Fields of Study

Major Field: Computer Science and Engineering

Studies in:

Networking	Prof. Anish Arora
Software Systems	Prof. Feng Qin
Artificial Intelligence	Prof. Wei-Lun Chao

Table of Contents

Page

List of Tables

List of Figures

Figure **Page**

Chapter 1: Introduction

Wireless Mesh Networks (WMNs) have emerged as a practical wireless solution able to extend coverage and connectivity as well as increase the capacity of wireless access networks for delivering broadband Internet services. In the infrastructure of WMNs, access points (APs) provide access to clients by forwarding network traffic to routers in a multi-hop fashion until it reaches a gateway that bridges the last-mile networks to the wireless/wired backbones. In this architecture, each node (AP/router) can be equipped with multiple radios that are able to access multiple channels over a communication spectrum. In such multi-channel multi-radio (MCMR) networks, by using channels that cause no interference to each other, concurrent communications are feasible and can potentially improve the network capacity.

WMNs provide wide coverage areas, especially spanning tens to a hundred kilometers in rural areas, to mitigate the broadband digital divide and extend connectivity with lower costs of backhaul connections. In general, WMN development aims to provide cost-effective broadband connectivity, involving diverse concerns from backhaul connections (middle-mile networks) to local area networks (last-mile networks). First, although the deployment cost can be significantly reduced by using wireless backbones compared to the installation cost of wired cables (e.g., fiber optic), other infrastructure costs, such as the cost to build up towers for support line-of-sight propagation,

still dominate the expense for WMN development. One way of reducing the cost is to leverage pre-existing towers in the development areas, and then a cost optimization problem must be solved. Second, because of the scarce nature of wireless spectrum resources, network performance is significantly affected by interference and congestion. There still remain gaps to optimize the achievable capacity by minimizing the impact of interference, jamming, and bottlenecks over a limited set of communication channels.

In addition, the most noticeable cause of throughput-latency performance degradation in WMNs is mainly due to poorly planned wireless networks and the lack of resilience in communication protocols adapting to different network configurations. Furthermore, the dynamicity of wireless meshes requires a resilient design of solutions to adapt to changing states, including node mobility, outburst flow traffic, as well as variable interference in last-mile networks, and sparse spectrum access in middle-mile networks. Therefore, our thesis aims to construct a well-planned and optimized wireless network with throughput efficiency by solving network optimization problems and addressing issues of complexity, scalability, and generalizability. Specifically, all these optimization problems and research challenges in WMNs can be categorized as network planning, channel scheduling, and packet routing.

1.1 Application and Challenges

1.1.1 Network Planning

It determines the network topology and the infrastructure (e.g., the choice of wireless radios) to satisfy throughput constraints with a minimum construction cost.

Even as the demand for broadband escalates worldwide to support real-time Internet services such as streaming, distance learning, and telehealth, rural areas tend to be underserved, if not unserved. In the US alone, of the 24 million Americans lacking access to fixed broadband services of at least 25 Mbps/3 Mbps as of 2016, the vast majority live in rural areas: 31.4% of the population in rural areas versus 2.1% in urban areas. Moreover, while fiber optic deployment continues, its high cost implies that wireless backbones remain necessary to address the broadband digital divide for the foreseeable future.

Among the wireless approaches, wireless mesh networks offer a resilient alternative for spanning tens to a hundred kilometers in rural areas. They offer the potential of being low-cost, partly because they use unlicensed spectrum, unlike cellular networks. Therefore, IEEE 802.11-based long-distance wireless mesh networks have been widely deployed to fill rural broadband gaps worldwide, [18, 51]. Directional antennas of small beamwidth with WiFi radio are mounted on the top of towers of height sufficient for line-of-sight propagation over the links. TV White Space (TVWS) meshes are also starting to proliferate [1] as a low-cost alternative since their spectrum offers non-line-of-sight propagation over foliage and obstructions, as compared to the 2.4GHz and 5GHz spectrum used in many IEEE 802.11 meshes. This property allows TVWS to use lower-height towers than WiFi, which is particularly cost-efficient for hilly rural areas. Also, the IEEE 802.22 TVWS range is competitive with 802.11 long-distance wireless links: a base station can provide a transmission radius of up to 100 kilometers [23, 72].

A major challenge for wireless meshes is sustainability. The growth of demand for their services versus their profitability is a chicken-and-egg problem made harder

by a lack of networking planning tools that support incremental deployment. Part of the challenge is to contain the deployment cost, where tower costs can dominate the total cost. Towers are typically located at or near population hubs that need network connectivity, but suitable towers/structures are not easily available or afforded to providers. Tower cost is height dependent (say \$100-\$5000 for masts/towers from 10-45m). Communication equipment cost is also a factor: while WiFi equipment is comparatively cheap (say \$50 for WiFi p2p antennas), TVWS equipment is yet to enjoy economy of scale.

Wireless Mesh Planning and MNO. Network planning in wireless meshes can be divided into the *last-mile network optimization (LNO)* problem and the *middle-mile network optimization (MNO)* problem. Both are NP-hard [9, 67]. As depicted in Fig. 1.1, the former problem involves placing a minimum number of edge access terminals (i.e., gateways) to collect traffic flows from home APs while satisfying their QoS requirements. A polynomial time LNO approximation algorithm [9] exists that recursively calculates minimum weighted Dominating Sets (DS) while guaranteeing QoS requirements in each iteration.

MNO seeks to connect the edge-access terminals to core network service providers with minimal infrastructure cost while satisfying the throughput constraints of each edge-access terminal. The MNO problem was first formulated [67] as:

> *Given a set of terminals to be connected to a given landline (and a set of non-terminals with existing towers to be used as relay vertices), determine the network topology, the tower heights of terminals, antenna types, and orientations, transmit powers, and the route for each terminal to the landline, such that throughput constraints of terminals are satisfied and the total cost of towers and antennas is minimized.*

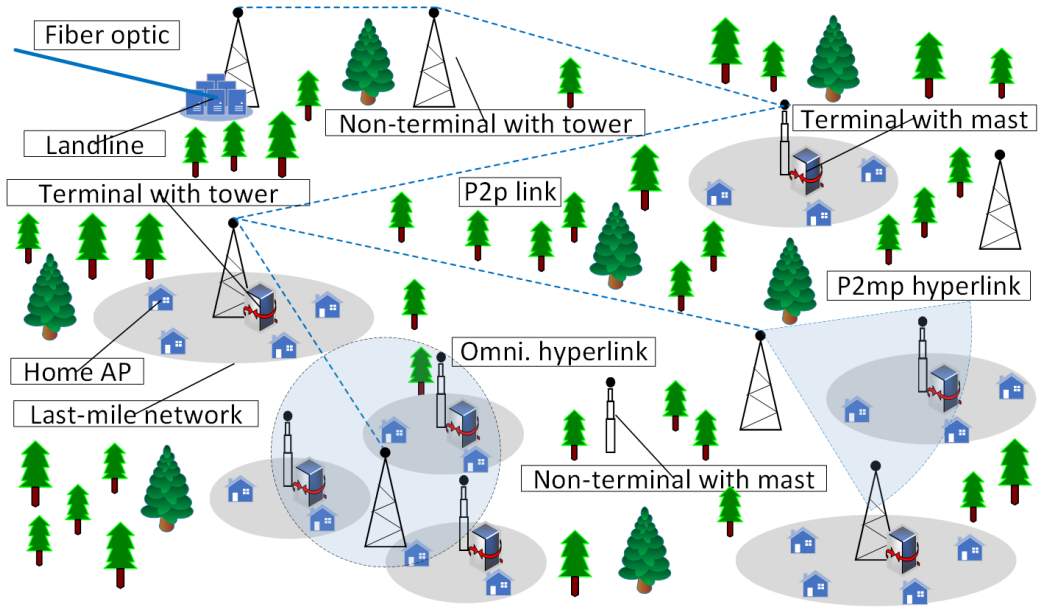


Figure 1.1: A rural middle-mile network connecting last-mile networks. Using Topology Construction of Minimum Steiner Tree (SteinerTC), we leverage non-terminals as relays to minimize tower heights of terminals and thus output a middle-mile network in the form of a minimal-cost Steiner tree. Based on the Steiner tree, using Capacitated Network Design (CND) we determine a minimum capacity installation conveying traffic from last-mile networks to the landline so that the infrastructure cost is (approximately) minimized.

MNO problem is not yet well solved: Its solution in [67] has exponential time complexity due to the formulation of height assignment and transmit power assignment as Linear Programming (LP) problems. Although a greedy approach has been proposed [61] for the NP-hard subproblem of topology construction along with tower height assignment, which has an approximation ratio of $O(\log n)$, we are not aware of approximation algorithms for MNO problem per se.

1.1.2 Channel Scheduling

Due to the scarcity of wireless channel resources, channel scheduling is crucial to minimize the impact of high end-to-end delay and the loss of achievable capacity caused by wireless interference and congestion. For reliably mitigating the effect of interference, frequency diversity (e.g., channel hopping) and time diversity (e.g., carrier sense multiple access) are conventionally leveraged to schedule communication channels.

Wireless mesh networks enable a cost-effective option for IoT connectivity in resource-, infrastructure- and access-limited settings, wherein device-to-device communications are achieved by multi-hop cooperation. The growing throughput-latency demands of 5G applications and the considerations of mobility and interference have motivated the development of performance-efficient networking platforms that can adapt to dynamic network environments. To that end, multi-channel multi-radio (MCMR) offers one such platform for high throughput-latency performance by increasing the degree of concurrent communication. This paper focuses on MCMR medium access control (MAC) protocols, which themselves play a key role in maximizing the performance in the presence of concurrent communication by scheduling transmissions in localities to avoid interference.

MCMR MACs have leveraged time diversity typically, i.e., using carrier sense multiple access with collision avoidance (CSMA/CA) to delay transmissions when needed or randomized transmission. In addition, frequency diversity, i.e., channel hopping or orthogonal frequency-division multiplexing, is applied to deal with interference. However, the state-of-the-practice channel scheduling of MCMR networks has several gaps in achieving high throughput-latency performance. This is especially true when

there are a number of concurrent traffic flows or when low latency is desired for bursty traffic.

As a representative example, let us consider the IEEE 802.15.4e Time Slotted Channel Hopping (TSCH) [2] protocol. TSCH has found significant adoption in industrial low-power applications of wireless sensor-actuator networks (WSAN) and multiple industrial standards [5, 3, 24]. As the demand for high data-rate services has grown, several TSCH-based protocols have been proposed for MCMR networks [11, 10].

The IEEE 802.15.4 standard leaves the choice of the channel sequence scheduler up to the protocol implementer. Scheduling may be realized in a centralized [60, 70] or distributed [6, 71] manner. However, channel hopping schedulers for TSCH typically rely on a globally fixed channel sequence. The sequence they use is relatively long, as the use of short sequences tends to be insufficient to mitigate interference, especially in the presence of concurrent traffic flows. The reliance on a large number of narrowband channels, in turn, yields a tradeoff with high spectrum utilization. Moreover, the fixed channel schedule underperforms in the presence of network dynamics. TSCH does not handle mobile scenarios effectively [59], as the channel sequence does not adapt to the changes in network interference with the change in network topology. TSCH is also vulnerable to jamming attacks [20, 21], as an attacker can reverse engineer the channel hopping sequence of a radio through observation of channel activities and then launch selective jamming attacks against all outgoing packets on the hopping sequence channels.

The throughput efficiency of a wireless mesh network with potentially malicious external or internal interference can be significantly improved by equipping nodes

with multi-radio access over multiple channels. However, multi-radio scheduling over a limited set of channels to minimize the effect of interference and maximize network performance in the presence of concurrent network flows remains a challenging problem. The state-of-the-practice in channel scheduling of multi-radios reveals not only gaps in achieving network capacity but also significant communication overhead.

1.1.3 Packet Routing

The adoption of multi-hop wireless mesh networks in access, backhaul, vehicular, swarm, or ad hoc contexts remained modest over the last couple of decades despite substantial R&D. However, recent trends are contributing to growth in the adoption of these networks. Among these trends is the growth in the number of edge nodes utilizing short-range communications—with high speed (i.e., mmWave or Sidelink) 5G communications [58, 25, 53] or low speed (i.e., LoRa or NB-IoT) communications in resource-limited and infrastructure-limited settings [69]. The increased availability of SDR (Software-defined Radio) (e.g., 5G-New Radio [82]) is contributing to this trend by providing greater interchangeability. Likewise, flexibility in the higher levels of the networking stacks (using, e.g., OpenFlow [52]) is contributing to this trend. Finally, most commercial networks, both realized and proposed, are full of fiber nearer the core of the network, but this architecture is not easily mimicked in military applications, resulting in an acute need for mesh technologies [66, 31].

A proper routing protocol carefully selecting forwarding relays between two ends enables detouring around holes and avoiding bottleneck nodes or jammed regions to maximize the throughput. Extant routing protocols often underperform in minimally

connected networks where holes are larger and more frequent. Minimal density networks are common in practice due to deployment cost constraints, mobility dynamics, and/or adversarial jamming. Protocols that use global search to determine optimal paths incur search overhead that limits scaling. Conversely, protocols that use local search tend to find approximately optimal paths at higher densities due to the existence of geometrically direct routes but underperform as the connectivity lowers and regional (or global) information is required to address holes.

One of the primary unresolved routing challenges for wireless meshes is the difference in handling the complexity of sparse network connectivity. Another primary challenge is related to the stability of paths, which is decreased by node mobility, changes in external interference (including adversarial jamming), and even changing traffic patterns. Therefore, designing a routing protocol to achieve high throughput-latency performance across a range of network densities, mobility, and interference dynamics remains challenging.

For providing high throughput and reliable communication over a variety of WMNs, the problems and challenges mentioned above need to be addressed appropriately. To that end, the proposed solution needs to generalize across a wide range of network configurations, in contrast to many extant routing protocols whose performance is tuned to specific configurations (e.g., high network density or low interference level). In particular, We also study machine learning techniques to improve the generalizability of routing protocols by leveraging the domain knowledge we have learned. This approach guides a possible and practical direction for achieving better performance and scalability.

Network Model

For the problems of channel scheduling and packet routing, without loss of generality, we assume that a wireless network is represented by a unit disk graph (UDG) $G = (V, E)$ whose nodes are uniformly randomly distributed over a 2-dimensional Euclidean plane. Each node $v \in V$ is associated with a unique ID and knows its global coordinates. For any nodes $v, u \in V$, edge $(v, u) \in E$ if and only if the Euclidean distance between v and u is not larger than the communication radius R. Let ρ denote the network density, where network density is defined to be the average number of nodes per R^2 area, and n is the number of nodes in V. It follows that all nodes in V are distributed in a square whose side is of length $\sqrt{\frac{n \times R^2}{\rho}}$.

1.1.4 Machine Learning for Generalized Routing

There has been considerable interest in machine learning to mimic the human ability to learn new concepts from just a few instances. While formal frameworks for machine learning, such as the language-identification-in-the-limit framework [28] and probably approximately correct (PAC) learning [79], have yet to match the high data efficiency of human learning, few-shot (or one-shot) learning methods [55, 83, 54] are attempting to bridge the gap by using prior knowledge to rapidly generalize to new instances given training on only a small number of samples with supervised information.

We explore the learnability of policies that generalize in the domain of graph routing, where a one-size-fits-all solution is challenging when the class of graphs has strict scalability limits for network capacity [87] or has significant graph dynamics [33, 29]. Manually designed algorithms and heuristics often cater to particular network

conditions and come with tradeoffs of complexity, scalability, and generalizability across diverse graphs. Many machine-learned algorithms in this space incur relatively high computational complexity during training [64], have high overhead at run-time, which limits their use in graphs with high dynamics or is applicable only for small-scale graphs or graphs of limited types (e.g., high-density graphs). Our work focuses attention on answering the following question:

> *Can we design a high data efficient machine learning algorithm for graph routing based on local search that addresses complexity, scalability and generalizability issues all at once?*

We answer this question in the affirmative for the all-pairs near-shortest path (APNSP) problem over the class of uniform random graphs in Euclidean and in hyperbolic metric spaces. It is well known that uniform random graphs in Euclidean metric spaces can represent the topologies inherent in wireless networks and that graphs in hyperbolic metric spaces can represent the tree-like topologies of the Internet and social networks where node degree follows a power-law distribution [14, 80]; the policies we learn are thus broadly applicable to real-world wireless and wired networks. Our key insight is that —in contrast to pure black-box approaches— domain knowledge suffices to theoretically guide the selection of "seed" graph(s) and corresponding sparse training data for efficiently learning models that generalize to (almost) all graphs in these geometric classes.

To motivate our focus on local search, we recall that approaches to solve the APNSP problem can be divided into two categories: global search and local search. Global search encodes the entire network state into graph embeddings [56] and finds optimal paths, whereas local search needs only node embeddings [30] to predict the next forwarder on the shortest path. The model complexity (in time and space)

resulting from the latter is inherently better than the former, as is the tolerance to network perturbations. The latter can even achieve stateless design, as is illustrated by geographic routing [16], where packet forwarding can be based on using only the location of the forwarding node and the destination. In other words, local search can achieve scalability and dynamic adaptation in a fashion that is relatively independent of the network configuration. We seek to mimic these properties by learning a low-complexity policy that can be instantiated and adapted in real time by virtue of its generalizability.

Of course, local search comes with the penalty of sub-optimality of the chosen path relative to the optimal (i.e., shortest) path. While this holds for both manual designs and machine-learned algorithms, the latter offers the potential for being better than the former by virtue of being rooted in optimization rather than in heuristics. Specifically, in our work, machine learning yields chosen paths that provably satisfy (with high probability) a reasonable bound on the achieved stretch over the shortest path; we refer to this as accurately finding a "near-shortest path".

1.2 Organization of This Thesis

This thesis presents our research for the WMN optimization problems to address the challenges and desiderata above. The document is organized as follows:

Chapter 2 discusses related work for the WMN optimization problems from network infrastructure development to communication schedule and routing path selection.

Chapter 3 introduces the middle-mile network optimization problem, which connects the last-mile networks in rural areas to the core network service providers for

delivering broadband connectivity in terms of topology construction, tower height assignment, antenna and orientation selection, as well as transmit power assignment. We provide the first polynomial time approximation solution for a generalized version (i.e., point-to-point links) of the middle-mile network optimization problem. Our solution also extends to hybrid networks, i.e., point-to-multipoint (i.e., WiFi p2mp) or omnidirectional (i.e., TV White Space), which can serve as hyperlinks in addition to point-to-point links to further reduce the cost of wireless links.

Chapter 4 proposes an adaptive channel hopping algorithm for multi-radio communication, QuickFire MAC (QF-MAC), that assigns per-node, per-flow "local" channel hopping sequences, using only one-hop neighborhood coordination. QF-MAC achieves a substantial enhancement of throughput and latency with low control overhead. QF-MAC also achieves robustness against network dynamics, i.e., mobility and external interference, and selective jamming attackers by dynamically adjusting channel sequences against interference.

Chapter 5 first shows a probabilistic bound that the search for the shortest path can be limited to an elliptic region whose size is a function of the network density and the Euclidean distance between the two endpoints. Then, we propose a geographic routing protocol, *QF-Geo*, which selects forwarding relays within the elliptic region, prioritizing those with sufficient capacity to avoid bottlenecks and can achieve high reliability and throughput-latency performance.

Chapter 6 proposes a simple algorithm that needs only a few data samples from a single graph for learning local routing policies that generalize across classes of geometric random graphs in Euclidean and hyperbolic metric spaces. We thus solve the all-pairs near-shortest path problem by training deep neural networks (DNNs) that

let each graph node efficiently and scalably route (i.e., forward) packets by considering only the node's state and the state of the neighboring nodes. Our algorithm design exploits network domain knowledge in selecting input features and selecting a "seed graph" and its data samples. The leverage of domain knowledge provides theoretical assurance that the seed graph and node subsampling suffice for learning that is generalizable, scalable, and efficient.

Finally, Chapter 7 summarizes the results of the thesis and points out the future research that can be based on this work.

Chapter 2: Related Work

2.1 Network Planning

2.1.1 Topology Construction (TC)

The TC problem is to determine a network topology that spans all the given terminals as well as a minimal height of the antenna tower of each terminal such that line-of-sight propagation using a WiFi p2p link exists for each edge. TC was formalized as follows [61]:

> *Given an undirected graph G = (V, E), obstruction locations and their heights on each edge in G, and a cost function of the tower height, determine a set of tower heights at vertices in V such that the subgraph induced by the edges covered by the set of tower heights is a subgraph (i.e., a spanning tree) of G spanning all vertices such that the total cost of the towers is minimized.*

The TC problem is known to be NP-hard and solvable by iteratively choosing the height increments of a vertex and a set of its neighbors with the best cost-to-benefit ratio until a spanning tree is determined. Note that this TC formalization only considers topologies that span all vertices, i.e., each vertex is terminal. One way of reducing the cost is to leverage pre-existing towers in rural areas to host non-terminal vertices (i.e., relays), thus eschewing additional tower costs while reducing the tower cost of some terminals. With this approach, the problem of determining the minimum

cost subgraph of G that connects all terminals and may include some non-terminals becomes that of computing the Minimum Steiner Tree. This leads us to pose the Topology Construction problem for the Minimum Steiner Tree (SteinerTC) problem of determining a set of tower height of terminals forming a Steiner tree with minimized cost.

2.1.2 Capacitated Network Design (CND)

The CND problem installs adequate capacity for meeting the bandwidth needs between a landline and edge access terminals. CND was formalized as follows [65, 32]:

> *Given an undirected graph G = (V, E), a set of terminals with their traffic demands, a landline, and uniform link capacity to be installed on edges, we are going to determine a minimum cost of capacity installation in G so that the installed capacity can route all traffic from terminals to the landline.*

The authors in [32] propose an approximation algorithm by introducing a relation between the CND problem and minimum Steiner tree problem. First, they take advantage of known solutions to Steiner tree problem [74] to obtain a Steiner tree with minimum cost. Second, their solution divides terminals into groups whose internal traffic flows are equal to or less than the cable capacity, and assign a hub in each group to forward the aggregated traffic from the corresponding terminals to the landline through the additionally installed cable capacity[1], if necessary, on the shortest path between the hub and landline. Since the CND problem satisfies throughput constraints by finding a minimum cost of wired cable installation in a given graph instead of considering a wireless case where line-of-sight propagation is necessary to support the transmission of WiFi spectrum, the CND problem is only a particular

[1]An edge in a graph can be installed more than one copy of cable.

instance of MNO problem. A solution to CND problem does not comprehensively solve MNO problem.

2.2 Channel Scheduling

2.2.1 Time Slotted Channel Hopping

TSCH is a channel hopping MAC protocol supported by IEEE 802.15.4e [2]. It inherits slotted time access from the IEEE 802.15.4 standard. To achieve reliable performance in the presence of interference and multipath fading, it leverages multi-channel communication and channel hopping based on a synchronized slotframe. Each slotframe is a collection of slots with a fixed length that repeats over time; every slot is long enough to transmit a data packet and receive an acknowledgment between a pair of nodes within some transmission radius.

Channel hopping in TSCH relies on channel offset schedules so as to avoid internal interference (and, in turn, communication collisions). The offset schedule is over a predefined sequence of channels, i.e., a globally shared sequence, instead of a randomly generated channel sequence. This is done primarily to reduce the channel synchronization overhead. The communication channel to be used for transmission and reception over a link at a slot follows the function CHS:

$$Channel = CHS[(ASN + OF)\ mod\ L], \tag{2.1}$$

where ASN stands for the absolute slot number, OF is the channel offset of a communication link between two nodes, and L denotes the length of a channel hopping sequence. The idea is to assign a different OF to links that can potentially interfere. Several heuristics [34] have been proposed to maximize the end-to-end reliability and minimize the end-to-end delay in TSCH MAC protocols, either by centralized or by

distributed scheduling of a node's transmission time slots (time diversity) and the channel offset in Eq. 2.1 (frequency diversity).

Even though TSCH-based protocols have shown good performance in industrial environments with stable topologies and low traffic, the reliance on a globally shared channel sequence limits their robustness to network dynamics. When L becomes small, its ability to avoid interference is negatively impacted. Furthermore, TSCH's channel hopping is vulnerable to a selective jamming attacker [20, 21], given its lack of channel adaption. Also, its use of a global channel hopping sequence inherently limits its resilience with respect to concurrent communications. The number of transmission links within an interference region must be at most L to avoid collision[2]. If L+1 links exist within the interference region, there exist at least two links that will collide predictably. In this case, the achievable capacity is reduced by at least $\frac{1}{L}$, which is more than the expected reduction when scheduling with random channel sequences.

2.2.2 Channel Hopping in Multi-channel Multi-radio Networks

Motivated by the spectrum efficiency gain they offer, multi-channel multi-radio (MCMR) mesh networks are increasingly being adopted in 5G contexts. The gain results primarily from the support of concurrent communications over orthogonal channels in MCMR networks. This support requires achieving rendezvous among communicating devices so that they are switched to the same channel at the same time. As illustrated by TSCH, channel hopping sequences provide a convenient basis for programming rendezvous synchronization, and so several channel hopping protocols for MCMR platforms have been proposed in the literature [47, 88, 78, 48, 17].

[2]We assume there is no repeated channel in a channel hopping sequence.

However, most extant work in MCMR MAC channel hopping schedulers only considers internal interference avoidance under static networks. Our solution further provides reliability and survivability to the MCMR networks with dynamics of external interference, adversarial jamming, and node mobility.

2.3 Packet Routing

2.3.1 Global Search Routing

Proactive algorithms tend to require global information [15], while reactive algorithms tend to invoke global search [8]. The extent to which non-local information is used or non-local activity is required is a primary limitation on scaling. As an aside, scaling is only challenging in the presence of changes in the network topology, the traffic, or the channel. As networks scale up, proactive algorithms tend to incur high control overhead, and reactive algorithms tend to have high setup latency.

2.3.2 Local Search Routing

Geographic routing protocols leverage geometric location information to forward data with low overhead [16]. As a simple policy option, greedy forwarding chooses the relay geographically closest to the destination. Yet, it may become trapped on the edge of a hole without any neighbor that is closer to the destination. The face routing algorithms [42, 38, 45] guarantee packet delivery by constructing a planar subset of the network and then walking along the faces of this planar graph that intersect with the line between the source and destination. Creating and maintaining a planar subset of the network is a proactive infrastructure that impacts the network capacity as the network scales. When the holes are frequent and large, the best path is likely to involve long detours, but the cost of discovering such a path through face routing

is high. Even more problematic, classical geographic routing rediscovers the route for every packet. Finally, geographic routing is typically not capacity aware, so its performance is impacted by its tendency to route too much traffic through hot spots. For low-density networks, the throttling caused by hot spots can substantially limit performance across a large portion of the network when traffic approaches capacity.

2.3.3 Machine Learning Techniques for Routing

Feature Selection for Routing

A classic feature for local routing comes from greedy forwarding [26], where the distance to the destination node (in a euclidean or hyperbolic metric space) is used to optimize forwarder selection. It has been proven that this feature achieves nearly optimal routing in diverse network configurations, including scale-free networks [41, 62]. A stretch bound on routing paths using greedy forwarding is investigated in diverse models with or without the assumption of unit disk graphs [27, 76, 77, 75, 84].

Other features for forwarder selection include Most Forward within Radius (MFR) [73], Nearest with Forwarding Progress (NFP) [35], the minimum angle between neighbor and destination (aka Compass Routing) [44], and Random Progress Forwarding (RPF) [57].

Network domain knowledge has also been used to guide search efficiency in routing protocols. A recent study [19] shows that searching for shortest paths in uniform random graphs can be restricted to an elliptic search region with high probability. A Stretch Factor at each node is used as an input feature used by its geographic routing protocol, *QF-Geo*, to determine whether a node's neighbors lie in the search region or not and to forward packets within a predicted elliptic rgion.

Generalizability of Machine Learned Routing

Only recently has machine learning research started to address generalizability in routing contexts. For instance, generalizability to multiple graph layout distributions, using knowledge distillation, has been studied for a capacitated vehicle routing problem [12]. Some of these explorations have considered local search. For instance, wireless network routing strategies via local search based on deep reinforcement learning [50, 49] have been shown to generalize to other networks of up to 100 nodes, in the presence of diverse dynamics including node mobility, traffic pattern, congestion, and network connectivity. Deep learning has also been leveraged for selecting an edge set for a well-known heuristic, Lin-Kernighan-Helsgaun (LKH), to solve the Traveling Salesman Problem (TSP) [86]. The learned model generalizes well for larger (albeit still modest) sized graphs and is useful for other network problems, including routing. Likewise, graph neural networks and learning for guided local search to select relevant edges have been shown to yield improved solutions to the TSP [36]. In related work, deep reinforcement learning has been used to iteratively guide the selection of the next solution for routing problems based on neighborhood search [85].

Chapter 3: Middle-mile Network Optimization

3.1 Problem Statement

In addition to satisfying the throughput constraints of terminals, the goal of the MNO problem is to minimize the cost of infrastructure deployment consisting of towers and antennas in a general way. In this section, we describe the problem and summarize the notations in Table 3.1.

3.1.1 Topology Construction of Minimum Steiner Tree (SteinerTC) Problem

Unlike the TC problem, SteinerTC, which leverages non-terminals (i.e., pre-existing towers) as relays to minimize the tower heights of terminals, is formulated as follows:

> *Given an undirected graph $G' = (V', E')$, a set of terminals $A \subset V'$, a set of non-terminals $B \subset V'$, $A \cup B = V'$, obstruction locations and their heights on each edge in G', and a cost function of the tower height, the goal is to determine a set of tower heights at vertices in A such that the subgraph induced by the edges covered by the set of tower heights in V' is a subgraph (i.e., a Steiner tree) of G' spanning all vertices in A such that the total cost of the towers at terminals is minimized.*

Table 3.1: Notations for Middle-mile Network Optimization

Symbol	**Meaning**
G	input graph of MNO problem $G = (V, E)$, $V = A \cup B$
A	set of terminals
B	set of non-terminals
$LN \in A$	landline (core network service provider)
$U \in \mathbb{N}$	capacity of wireless p2p link
$cTower$	tower cost function $cTower : [HTMIN, HTMAX] \to \mathbb{R}$
$HTMIN \in \mathbb{R}$	minimum tower height
$HTMAX \in \mathbb{R}$	maximum tower height
$cLink$	link cost function $cLink : \mathbb{N} \to \mathbb{R}$
dem	throughput demand function $dem : V \to [0, U]$, $\forall v \in B, dem(v) = 0$
$R \in \mathbb{R}$	maximal transmit distance of WiFi p2p links
ob	normalized obstruction height function $ob : E \to \mathbb{R}$
$G_{SteinerTC}$	output graph of MNO problem $G_{SteinerTC} = (V_{SteinerTC}, E_{SteinerTC})$ $A \subseteq V_{SteinerTC} \subseteq V$
h	tower height function $h : V \to [HTMIN, HTMAX]$ $\forall v \in B$, $h(v)$ is given and fixed
rt	route function from terminals to LN $rt : V \to P$ P: set of paths composed of edges in $E_{SteinerTC}$
f	traffic flow function $f : E_{SteinerTC} \to \mathbb{R}$
$linkset_{MP}$	hyperlink function at vertices with MP antennas $linkset_{MP} : V_{SteinerTC} \to CONFS$ $CONFS$: set of subset of MP antenna configurations (see Algorithm 3)
$linkset_{Omni}$	hyperlink function at vertices with $Omni$ antennas $linkset_{Omni} : V_{SteinerTC} \to VS$ VS: set of subsets of $V_{SteinerTC}$

Different from the output graph of the TC problem, which yields a minimum cost spanning tree where the induced vertices are all terminals, the output graph of SteinerTC is a minimum cost Steiner tree where the spanned vertices consist of all terminals and some of the non-terminals.

The following statement shows the hardness of SteinerTC, which is proved by reducing TC to SteinerTC.

Theorem 1. *SteinerTC is at least as hard as TC, which is an NP-hard problem.*

Proof. For any two adjacent vertices u and v in graph G in the TC problem, assume it is allowed to create at least one intermediate vertex with a fixed tower height between u and v but zero cost if there exists an edge (u, v) of line-of-sight propagation associated with height increments for u and v, denoted as $h^+(u)$ and $h^+(v)$ respectively. Accordingly, based on $h^+(u)$ and $h^+(v)$ which make edge (u, v) with cost increment $cTower^+(h^+(u))$ and $cTower^+(h^+(v))$ in TC problem, it is able to create one or more intermediate vertices $i_1, \ldots, i_r$ with fixed heights $h(i_1), \ldots, h(i_r)$ which make the same cost increment, $cTower^+(h^+(u)') + cTower^+(h^+(v)') = cTower^+(h^+(u)) + cTower^+(h^+(v))$, to make edges (u, i_1) and (i_r, v) of line-of-sight propagation with heights $h^+(u)'$ and $h^+(v)'$.

Since $h(i_1), \ldots, h(i_r)$ are sufficient to make line-of-sight propagation over edges $(i_1, i_2), \ldots, (i_{r-1}, i_r)$, we can say that for any new paths between (u, v), there exists a path with the minimal cost equal to the cost of $cTower^+(h(u)) + cTower^+(h(v))$. Some intermediate vertices with fixed tower heights could be reused as relays in a new path between any two neighbor vertices in G. Thus the degree of intermediate vertices is at least two.

Therefore, for an instance of SteinerTC problem with the graph $G' = (V', E')$, there exists an instance of TC problem with the graph $G = (V, E)$, which can be mapped to G', where all the intermediate vertices are non-terminals in G'. □

3.1.2 Reformulation of MNO

Recall that the parameters to be determined in the MNO problem are network topology, tower heights, antenna type and orientations, transmit powers, and traffic routes. The SteinerTC problem—finding a minimum cost Steiner tree by identifying a set of tower heights—determines the first two parameters. Given this Steiner tree as an input graph, the CND problem finds a minimum capacity installation routing traffic to the landline. It satisfies throughput constraints in consideration of line-of-sight propagation over the WiFi spectrum. Note that, in a homogeneous case (p2p links only) of the MNO problem, the parameters of antenna type, orientations, and transmit powers can be determined based on the edges in the Steiner tree. (In hybrid networks, the schemes of assigning antenna orientations of WiFi p2mp and transmit power are presented in Section 3.3.) Hence, solving SteinerTC and CND problems in sequence suffices to determine the parameters in the MNO problem.

> ***Input***. An undirected graph $G = (V, E)$, a set $A \subset V$ of terminals and a set $B \subset V$ of non-terminals, $A \cup B = V$, a landline $LN \in A$, a cost function of tower height $cTower$, a cost function of link installation $cLink$, the link capacity U, and throughput demand $dem(v) \leq U$ for each terminal.

Note that for any two vertices u and v, $(u, v) \in E$, $dist(u, v) \leq R$. For each edge $(u, v) \in E$, a normalized obstruction height $ob(u, v)$ exists in the middle of edge (u, v). Line-of-sight propagation is available over a wireless p2p link connecting u and v if the sum of the tower heights at u and v is not less than $2 \times ob(u, v)$.

Output. A connected graph $G_{SteinerTC} = (V_{SteinerTC},\ E_{SteinerTC})$, along with a height function h for each terminal in A, the route function rt for each terminal in V indicating the routing path from terminals to the landline, the traffic flow function f for each edge in $E_{SteinerTC}$, and the hyperlink functions $linkset_{MP}$ and $linkset_{Omni}$ if a hybrid network is constructed, such that throughput demands of terminals are satisfied and the cost of tower and wireless link installation $\sum_{v \in A} cTower(h(v)) + \sum_{e \in E_{SteinerTC}} cLink(\lceil \frac{f(e)}{U} \rceil)$ is minimized.

We note that $linkset_{MP}$ and $linkset_{Omni}$ denote the set of hyperlinks. For vertices v where p2mp antenna(s) are installed, $linkset_{MP}$ specifies the p2mp antenna configurations at v that include the covered vertices and effective direction/range. For vertices v where an omnidirectional antenna is installed, $linkset_{Omni}$ returns vertices in v's neighborhood connected to the omnidirectional antenna at v.

3.2 Polynomial Time Approximation Algorithm for SteinerTC Problem

3.2.1 Solution to SteinerTC Problem

A greedy algorithm, TC-ALGO, is presented in [61] to solve TC with a logarithmic cost bound. In each iteration of TC-ALGO, all terminals are considered with different height increments associated with cost increments, using a doubling search to find the one with the lowest cost-to-benefit ratio. For each combination of a terminal and a particular height increment considered, TC-ALGO considers its neighbor terminals with the minimum height increments to achieve line-of-sight connections by calling the subroutine of STAR-TC-ALGO. Its evaluation metric is the cost-to-benefit ratio, defined as the minimum increment in cost for the maximum reduction in the number of isolated components. Note that a component of a graph denotes a maximal connected subgraph.

Algorithm 1: STAR-SteinerTC-ALGO

Input: $G = (V, E)$, terminal set A, non-terminal set B, height function h, vertex v, height increment δ at v
Output: cost-benefit-ratio r'_{best}, height increment function $incr$
$cTower^{+}(h(v)) := cTower(h(v) + \delta)$ - $cTower(h(v))$;
$nbr_{logic} := \{u \mid v,\ u \in A,\ u$ is not in the same component as v in $cover(h)$, there exists a path $p = (v, v_1, \ldots, v_r, u)$ from v to u in G s.t. $|p| \geq 2$, $v_1, \ldots, v_r \in B$, heights $(h(v) + \delta)$ at v and heights $h(v_1), \ldots, h(v_r)$ cover edges $(v, v_1), \ldots, (v_{r-1}, v_r)\}$;
for *each vertex* $u \in nbr_{logic}$ **do**
    if $(u, v) \in E$ **then**
        $h^{+}(u) :=$ smallest β s.t. heights $(h(v) + \delta)$ at v and $(h(u) + \beta)$ at u cover edge (u, v);
    else
        search the path $p_{best} = (v, v_1, \ldots, v_s, u)$ from v to u with lowest cost increment;
        $h^{+}(u) :=$ smallest β s.t. heights $(h(v) + \delta)$ at v and $(h(u) + \beta)$ at u cover path $p_{best} = (v, v_1, \ldots, v_s, u)$;
    end
    $cTower^{+}(h(u)) := cTower(h(u) + h^{+}(u))$ - $cTower(h(u))$;
end
$L :=$ list of vertices in nbr_{logic} in ascending order of $cTower^{+}$;
for *each component* $D \in cover(h)$ **do**
    remove from L all vertex $u \in D \cap nbr_{logic}$ except the one with lowest $cTower^{+}$;
end
$r'_{best} := \infty$; $k_{best} := 0$;
for $1 \leq k \leq |L|$ **do**
    $r'_{tmp} := \frac{cTower^{+}(h(v)) + \sum_{i=1}^{k} cTower^{+}(h(L[i]))}{k}$;
    if $r'_{tmp} < r'_{best}$ **then**
        $k_{best} := k$; $r'_{best} := r'_{tmp}$;
    end
end
for $u \in V$ **do**
    $incr(u) := 0$;
end
for $u \in L[1 \ldots k_{best}]$ **do**
    $incr(u) := h^{+}(u)$;
end
$incr(v) := \delta$;
return $(r'_{best}, incr)$;

Given a terminal v with a height increment $h^+(v)$, STAR-TC-ALGO sorts the terminal's neighbors $u_1, \ldots, u_s$ by the cost of the neighbors' minimum height increments $cTower^+(h(u_1)), \ldots, cTower^+(h(u_s))$, that cover edges $(v, u_1), \ldots, (v, u_s)$. Then it returns a list of selected neighbors achieving the lowest cost-to-benefit ratio associated with v and $h^+(v)$. In each iteration of TC-ALGO, the terminal with the height increment and some of its neighbors with minimum sufficient height increments achieving the lowest cost-to-benefit ratio over all combinations are selected to add to h. TC-ALGO iteratively chooses the height increments until the number of components is reduced to one. Note that the benefit, standing for the number of reduced components, is at least one. It guarantees the progress of component reduction in each iteration. Moreover, components covered by the height function h are denoted by $cover(h)$. Initially, $cover(h)$ stands for all isolated terminals due to zero height increment at all terminals.

In the TC problem, all vertices are considered terminals and assigned tower heights. However, in the SteinerTC problem, some of the vertices considered non-terminals with fixed tower heights can work as relays between two terminals to reduce the tower heights at terminals if applicable. To find the lowest cost-to-benefit ratio in the SteinerTC problem for a given terminal v with a height increment, we search not only the adjacent terminals of v but the terminal u that can be connected through some non-terminals $v_1, \ldots, v_r$ by the path $(v, v_1, \ldots, v_r, u)$. We use the term of *logical neighbor* of a given vertex v to denote a terminal u that can be connected by a single edge (u, v) or a path $(v, v_1, \ldots, v_r, u)$ where only v and u are terminals.

STAR-SteinerTC-ALGO, shown in Algorithm 1, is adapted from STAR-TC-ALGO. (Note that the main greedy routine, SteinerTC-ALGO, to SteinerTC problem remains

the same as TC-ALGO.[3]) In STAR-SteinerTC-ALGO, line 4 collects the logical neighbors of v that are not in the same component as v in $cover(h)$, since the benefit of v cannot be improved by increasing the height of the vertices in the same component of v. The for loop from lines 5 to 13 determines the smallest height increment of each vertex u in nbr_{logic} that makes u connected to v. If there is a single edge (u, v) between u and v, we directly assign the smallest height increment at u to cover this edge. Otherwise, we search a minimum cost path[4] from v to u including only non-terminals, $v_1, \ldots, v_s$, except u and v and then assign the smallest height increment at u to cover (v_s, u). Note that the tower heights at $v_1, \ldots, v_s$ must be high enough to cover edges $(v, v_1), \ldots, (v_{s-1}, v_s)$. A vertex with the lowest cost increment (as well as the lowest height increment) is chosen from the same component, and its vertices are sorted in ascending order of the cost increment from lines 14 to 17. Note also that each element in L reduces the number of components in $cover(h)$ by exactly one. Then, we can find the lowest cost-to-benefit ratio by selecting the first k elements in L. The for loop from lines 19 to 24 determines the size of k to obtain the lowest cost-to-benefit ratio r'_{best}. The remaining lines add the corresponding height increments for v and the chosen logical neighbors of v to the height increment function $incr$ and then return it.

The performance bound of our greedy algorithm is shown in Theorem 2, whose proof is relegated to Appendix A.

[3]In each iteration, the greedy algorithm of SteinerTC-ALGO considers different height increments only for terminals since the tower heights of non-terminals are fixed.

[4]The minimum cost path between v and u can be found in polynomial time since the heights of v and all non-terminals are given.

Theorem 2. *The tower cost associated with the height function h determined by SteinerTC-ALGO is at most* $2\ln|A|OPT$.

3.2.2 Solution to CND Problem

The approximation algorithm presented in [32] installs capacity per a given minimum cost Steiner tree to route the traffic flows from terminals to the landline. The algorithm first partitions terminals into groups according to their throughput demands. Note that the total aggregated traffic in a group is upper-bounded by the cable capacity. For each group, it chooses the vertex with the shortest path to the landline as a hub. The algorithm of the grouping guarantees that any internal aggregated traffic in a group does not exceed $U - dem(v_h)$, where U is the uniform cable capacity and v_h is the hub in this group. Thus, installing only one cable over every edge in the minimum cost Steiner tree can support the traffic flows from terminals to hubs. This capacity installation makes connectivity in the Steiner tree.

Moreover, one cable is installed on each edge for each group over the shortest path from its hub to the landline if additional capacity is needed. This capacity installation guarantees the bandwidth requirement from hubs to the landline. Hence, the capacity installation from terminals to hubs and from hubs to the landline satisfies the throughput constraints. Note that the cost of capacity installation from terminals to the landline is dominated by their hop distances to the landline. In our solution to the CND problem, we take advantage of the grouping scheme in [32] on the minimum cost Steiner tree, $G_{SteinerTC}$, determined by SteinerTC-ALGO. Each group selects the vertex with the shortest hop distance to the landline LN as a hub.

For the capacity installation, we first install one wireless link per edge in $G_{SteinerTC}$. Second, to route the aggregated traffic of groups to LN, one additional wireless link, if necessary, is installed per edge on the paths from hubs to LN, which are formed by the corresponding edges in $G_{SteinerTC}$. In our solution, the route function rt and the traffic flow function f are determined by the grouping scheme and the terminal demands.

3.2.3 Performance Analysis

In this section, we will derive the cost performance ratio of our approximation algorithm to MNO problem by analyzing the total cost of tower and wireless link installation ($\sum_{v \in A} cTower(h(v)) + \sum_{e \in E_{SteinerTC}} cLink(\lceil \frac{f(e)}{U} \rceil)$). In the following, let $cTower(SteinerTC)$ and $cLink(CND)$ respectively denote the tower cost and wireless link cost in our solution.

First, we analyze the cost of tower deployment. Let $cTower(OPT)$ be the cost of towers in the optimal solution to the SteinerTC problem. Suppose there exists a minimum cost spanning tree where all vertices are terminals. The tower cost of this spanning tree is denoted by $cTower(MST)$. Since our solution to the SteinerTC problem exploits non-terminals with existing towers to further minimize the cost of towers, we note that $cTower(SteinerTC) \leq cTower(MST)$. In addition, since our solution to the SteinerTC problem has the cost bound ratio of $2 \ln |A|$ to the optimal solution, we get the equation:

$$cTower(OPT) \leq cTower(SteinerTC) \leq 2 \ln |A| cTower(OPT) \tag{3.1}$$

Second, we analyze the cost of wireless link installation. Let $cLink(OPT)$ be the cost of all wireless links in the optimal solution to the CND problem. Since

$cLink(CND)$ consists of the cost of installed links from terminals to hubs and the cost of installed links from hubs to the landline, we use $cLink(CND_{T2H})$ and $cLink(CND_{H2L})$ to denote the two costs, respectively. Let $cLink(MST_{T2H})$ be the cost of link installation from terminals to hubs in MST where each edge connects two terminals and the number of edges, $|A|-1$, is minimal. Thus, we know $cLink(MST_{T2H}) \leq cLink(OPT)$ and $cLink(MST_{T2H}) \leq cLink(CND_{T2H})$. Since the worst case of the number of edges in a Steiner tree is the number of all vertices minus one $(|A|+|B|-1)$, we get the worst-case bound of $cLink(CND_{T2H})$: $cLink(MST_{T2H}) \leq cLink(CND_{T2H}) \leq \frac{|A|+|B|-1}{|A|-1} cLink(MST_{T2H})$. Also, we infer the bound ratio of $cLink(CND_{T2H})$ to $cLink(OPT)$:

$$\begin{aligned} & cLink(CND_{T2H}) \leq \tfrac{|A|+|B|-1}{|A|-1} cLink(MST_{T2H}) \\ & \leq \tfrac{|A|+|B|-1}{|A|-1} cLink(OPT) = (1+\tfrac{|B|}{|A|-1}) cLink(OPT) \end{aligned} \tag{3.2}$$

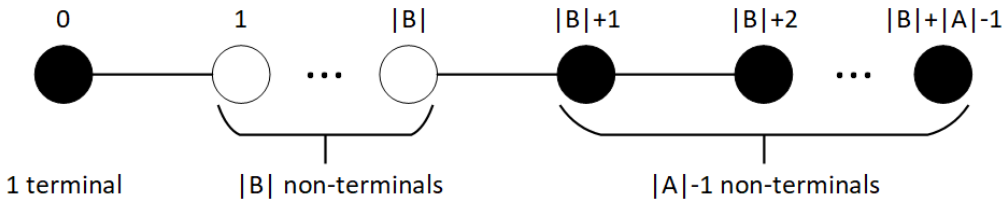


Figure 3.1: Worst-case topology of CND solution.

Moreover, the upper bound of $cLink(CND_{H2L})$ is obtained by considering the worst case of total hop distance from hubs to the landline. Let all terminals have a uniform demand, and all wireless links have a uniform link capacity. Let γ denote the ratio of $\frac{link\ capacity}{terminal\ demand}$. To calculate the lower bound of $cLink(OPT)$, we consider the best case of the topology, where all the vertices have a one-hop distance to the landline. In that case, a wireless link is installed on every edge from terminals to the

landline. Thus, we have the best-case bound of $cLink(OPT)$, $link\ cost \times (|A|-1) \leq cLink(OPT)$. Now we calculate the worst case of the topology that is a long-chain topology where the 0th vertex is the landline (terminal), the 1st to the $(|B|)$-th vertices are non-terminals and the $(|B|+1)$-th to $(|B|+|A|-1)$-th vertices are the remaining $|A|-1$ terminals. As Fig. 3.1 depicted, the numbers labeled at the top stand for the hop distances for the vertices to the landline. Note that the grouping scheme to CND problem separates all terminals vertex into $\frac{|A| \times terminal\ demand}{link\ capacity} = \frac{|A|}{\gamma}$ groups. For convenience, $\frac{|A|}{\gamma}$ is denoted as m. For each group, we select a hub with a minimum hop distance to send the aggregated traffic equal to (or less than) the link capacity to the landline. Thus, in the 1st group containing terminals with the labels, $0, |B|+1, \ldots, |B|+\gamma-1$, we select vertex 0 as a hub. In the 2nd group containing terminals with the labels, $|B|+\gamma, |B|+\gamma+1, \ldots, |B|+2\gamma-1$, we select vertex $|B|+\gamma$ as a hub. In the last group containing terminals with the labels, $B+(m-1)\gamma, B+(m-1)\gamma+1, \ldots, B+m\gamma-1)$, we select vertex $B+(m-1)\gamma$ as a hub. Now, the total hop distance from hubs to the landline in the worst case can be presented as follows:

$$
\begin{aligned}
& 0+(|B|+\gamma)+\cdots+(|B|+(m-1)\gamma) \\
&= \gamma[(m-1)\tfrac{|B|}{\gamma} + \tfrac{m(m-1)}{2}] \\
&= (m-1)(|B| + \tfrac{m\gamma}{2}) \\
&= (\tfrac{|A|}{\gamma}-1)(|B| + \tfrac{|A|}{2}), m \leftarrow \tfrac{|A|}{\gamma} \\
&\leq \tfrac{|A|}{\gamma}(|B| + \tfrac{|A|}{2}) = \tfrac{|A|^2+2|A||B|}{2\gamma}
\end{aligned}
$$

According to the best-case bound of $cLink(OPT)$ and the worst-case bound of $cLink(CND_{H2L})$, we know $link\ cost \times (|A|-1) \leq cLink(OPT)$ and $link\ cost \times (|A|-1) \leq cLink(CND_{H2L}) \leq link\ cost \times \frac{|A|^2+2|A||B|}{2\gamma}$. Hence, the performance ratio of $cLink(CND_{H2L})$ to $cLink(OPT)$

can be stated as follows:

$$\begin{aligned} cLink(CND_{H2L}) &\leq \tfrac{|A|^2+2|A||B|}{2\gamma(|A|-1)} cLink(OPT) \\ &\leq \tfrac{|A|^2+2|A||B|}{\gamma|A|} cLink(OPT) \leq \tfrac{|A|+2|B|}{\gamma} cLink(OPT) \end{aligned} \tag{3.3}$$

Note that $|A| - 1 \geq \frac{|A|}{2}$ due to $|A| \geq 2$.

Finally, we calculate the cost performance ratio of our approximation algorithm to the MNO problem. Let C_{OPT} be the total infrastructure cost of the optimal solution to the MNO problem. Since the sum of the costs of the respective optimal solutions in SteinerTC and CND problems is the cost lower bound for C_{OPT}, we know $cTower(OPT) + cLink(OPT) \leq C_{OPT}$. Let C_{ALGO} be the total infrastructure cost of our solution to the MNO problem, where $C_{ALGO} = cTower(SteinerTC) + cLink(CND_{T2H}) + cLink(CND_{H2L})$. Now, we are going to discuss the three following cases of terminal demand and articulate their performance ratio.

Case 1: The total demand from terminals is not larger than the link capacity. In this case, $C_{ALGO} = cTower(SteinerTC) + cLink(CND_{T2H})$ since the installation of a single wireless link on each edge from terminals to hubs is sufficient to convey the total traffic from all terminals. According to (3.1) and (3.2), we have $cTower(SteinerTC) \leq 2\ln|A| cTower(OPT) \leq 2\ln|A|)C_{OPT}$ and $cLink(CND_{T2H}) \leq (1+\frac{|B|}{|A|-1})cLink(OPT) \leq (1 + \frac{|B|}{|A|-1})C_{OPT}$. Hence, the cost performance ratio of case 1 is

$$\frac{C_{ALGO}}{C_{OPT}} \leq (1 + 2\ln|A| + \tfrac{|B|}{|A|-1}) \tag{3.4}$$

Case 2: The total demand from terminals is larger than link capacity and the terminal demands are uniform. In this case, $C_{ALGO} = cTower(SteinerTC)+cLink(CND_{T2H})+cLink(CND_{H2L})$. According to (3.3), we know $cLink(CND_{H2L}) \leq \frac{|A|+2|B|}{\gamma} cLink(OPT) \leq \frac{|A|+2|B|}{\gamma} C_{OPT}$. Hence, the cost performance ratio of case 2 is

$$\frac{C_{ALGO}}{C_{OPT}} \leq (1 + 2\ln|A| + \tfrac{|B|}{|A|-1} + \tfrac{|A|+2|B|}{\gamma}) = O(\ln|A| + \tfrac{|B|}{|A|-1} + \tfrac{|A|+|B|}{\gamma}) \tag{3.5}$$

Case 3: The total demand from terminals is larger than the link capacity, and the terminal demands are non-uniform. In this case, we can reuse the result from (3.5) by letting γ be $\frac{link\ capacity}{maximum\ demand}$.

According to the evaluation of our solution's worst-case cost bound, the cost performance ratio bound is shown in Theorem 3.

Theorem 3. *Our solution has a cost performance ratio of* $O(\ln|A| + \frac{|B|}{|A|-1} + \frac{|A|+|B|}{\gamma})$, *where A and B respectively denote the number of terminals and non-terminals and* γ *is the ratio of* $\frac{link\ capacity}{terminal\ demand}$.

3.3 Cost Minimization Heuristic for Hybrid Mesh Networks

In this section, we extend the algorithm for solving the MNO problem in hybrid networks that allow the incorporation of point-to-multipoint (i.e., WiFi p2mp) and omnidirectional links (i.e., TVWS) as hyperlinks. We present a heuristic for adding p2mp or omnidirectional hyperlinks (i.e., WiFi for p2mp or TVWS for omnidirectional) if and only if they reduce the total cost while preserving the satisfaction of throughput constraints. The algorithm iteratively finds a vertex that has some links with residual capacity and then replaces those links with a hyperlink while considering cost reduction. Below, we first articulate the concept of link replacement. Second, we present a replacement scheme for the case of p2mp hyperlinks that substitute p2p links without changing the topology $G_{SteinerTC}$, tower heights, and the traffic routing. Third, we introduce a replacement scheme for the case of omnidirectional hyperlinks that substitute some p2p links and for altering the corresponding tower

heights adapting to omnidirectional antennas and their subordinate directional antennas when that can offer cost reduction. Finally, we discuss the assignment of transmit power in hybrid networks.

Table 3.2: Notations for Cost Minimization in Hybrid Networks

Symbol	**Meaning**
U	p2p link / p2mp hyperlink capacity
R_{MP}	maximal transmit distance of MP antenna, $R_{MP} \leq R$
BW	tunable beamwidth of MP antenna, $BW \leq BWMAX$
$BWMAX$	maximal beamwidth of MP antenna
RAD_{MP}	tunable radius of MP antenna, $RAD_{MP} \leq R_{MP}$
X	set of vertices covered by a hyperlink
$cAntenna$	antenna cost function $cAntenna : \{PP, MP, Omni, OmniSD\} \rightarrow \mathbb{R}$
rng_{MP}	MP range verification function $rng_{MP} : E_{SteinerTC} \times V_{SteinerTC} \times V_{SteinerTC}$ $\times \mathbb{R} \times \mathbb{R} \rightarrow \{0,1\}$
U_{Omni}	omnidirectional hyperlink capacity
R_{Omni}	maximal transmit distance of omnidirectional antenna
RAD_{Omni}	radius of omnidirectional antenna, $RAD_{Omni} \leq R_{Omni}$
$HTOmni$	fixed tower height of omnidirectional antenna
$HTOmniSD$	fixed tower height of subordinate directional antenna of omnidirectional hyperlink
rng_{Omni}	omnidirectional range verification function $rng_{Omni} : V_{SteinerTC} \times \mathbb{R} \times V_{SteinerTC} \times \mathbb{R} \rightarrow \{0,1\}$
V_{Omni}	set of vertices with omnidirectional antennas
$V_{OmniSD}(v)$	set of vertices only covered by the omnidirectional hyperlink induced from v

3.3.1 Hyperlink Replacement Strategy

The solution to the CND problem we have adopted involves partitioning, wherein terminals are divided into groups whose aggregated traffic is at most the link capacity. This scheme results in links within groups that are not fully utilized. This underutilized wireless link capacity is called *residual capacity*. For the traffic flow function f returned from the MNO problem, it computes the required traffic for each edge $e \in E_{SteinerTC}$. Let U be the link capacity over its spectrum. We assume that a p2p link (which has two p2p antennas) and a p2mp hyperlink (which has one p2mp antenna and one or more p2p antennas) have the same capacity U. However, an omnidirectional hyperlink (which has one omnidirectional antenna and one or more of its subordinate directional antennas) can have a different capacity U_{Omni}. The residual capacity of edge e is thus defined as $U\lceil\frac{f(e)}{U}\rceil - f(e)$.

We now describe our cost minimization heuristic that replaces underutilized edges with p2mp and omnidirectional hyperlinks. Note that one edge is installed with one or more p2p links. The related notations are listed in Table 3.2. Recall that although the primary goal of hyperlink replacement is to minimize the infrastructure cost in hybrid networks, the replacement must satisfy the throughput constraints. The heuristic does not change the MNO solution topology or the traffic routing since changing the topology will alter the traffic flows, which may break the constraints guaranteed by solving CND. In other words, replacement with a p2mp / omnidirectional hyperlink only happens at "candidate" vertices concerning some/all of their respective neighbors; it does not include any vertex at a distance of more than one hop from any candidate vertex.

More specifically, given a candidate vertex v, the replacement only involves the edges connected to its children instead of the edge $(v, v.parent)$ that conveys the aggregated traffic of all its children. We choose not to deploy a hyperlink that covers $(v, v.parent)$ and the edges from v to its children. The reasons for this are: First, for p2mp antennas, replacing $(v, v.parent)$ may undermine throughput performance because only one link in a p2mp hyperlink can transmit at a given time. Second, for omnidirectional links, such replacement occupies twice the capacity over the spectrum but does not effectively reduce the cost of tower height, which is the dominant factor for infrastructure cost. At v, we apply its height to *HTOmni*, which may increase cost ($h(v) < HTOmni$) or reduce cost ($h(v) > HTOmni$). Nevertheless, at vertex $v.parent$, we apply its height to $max(h(v.parent), HTOmniSD)$ because $v.parent$ still has other induced edges(s). Note that $h(v)$ is likely larger than *HTOmniSD* (10m) and the available channels in the omnidirectional link's spectrum are limited and location-dependent. Therefore, allowing replacement only at child vertices offers the chance to replace more p2p links and reduce the corresponding tower heights.

3.3.2 MP Deployment

For a p2mp hyperlink, we use one p2mp antenna to share the capacity U among its corresponding p2p antennas. A p2mp antenna is modeled by direction, beamwidth, and radius. We denote, for a p2mp antenna deployed at v, the direction of the antenna pointed towards vertex u as $\overrightarrow{vu}$. We assume the p2mp antenna is flexible to configure its beamwidth and transmit radius so that one p2mp antenna can be set to different sector ranges; this is typical of today's p2mp antennas.

For MP deployment, four constraints must be guaranteed if and only if a p2mp hyperlink is deployed at v to replace p2p links associated with the edges $(v, x), x \in X$, where X denotes the set of selected children of v.

$$
\begin{aligned}
&Cost : cAntenna(MP) < cAntenna(PP)|X| \\
&Capacity : \textstyle\sum_{x \in X} f((v, x)) \leq U \\
&Range : \textstyle\prod_{(v,x) \in E_{SteinerTC}, x \in X} rng_{MP}((v, x), v, u, BW, \\
&\qquad RAD_{MP}) = 1 \\
&Interference : \textstyle\sum_{e \in E_{SteinerTC} \setminus \{(v,x)|x \in X\}} rng_{MP}(e, v, u, \\
&\qquad\qquad BW, RAD_{MP}) = 0
\end{aligned}
$$

Cost and capacity constraints ensure that replacement reduces total costs and satisfies the demand for throughput. Note that the tower heights at terminals do not change in MP deployment. The range verification function rng_{MP} examines if the edge segment (v, x) is covered by the p2mp antenna at v and pointed towards u, with the beamwidth BW and radius RAD_{MP}. The interference constraint guarantees that the deployed p2mp antenna does not affect any edge other than $(v, x), x \in X$.

We now state the cost minimization problem of hybrid MP deployment as follows[5]:

> ***Input***. *$G_{SteinerTC} = (V_{SteinerTC}, E_{SteinerTC})$, the corresponding traffic flows on edges in $E_{SteinerTC}$.*
> ***Output***. *A set of vertices where p2mp antennas are deployed, along with direction, beamwidth, and radius assignments, such that all four constraints are satisfied and the cost of wireless links is minimized.*

The solution in Algorithm 2 considers candidate vertices (and their respective children) in a bottom-up fashion. Each candidate vertex performs local MP deployment with the greedy Algorithm 3 that selects the antenna configuration (direction, beamwidth, radius) containing the largest number of uncovered vertices at each iteration.

[5]Note that we can maximize the cost reduction by maximizing the number of replaced p2p links. The problem can be considered a maximum coverage problem [22].

Algorithm 2: MP-DEPLOY-ALGO

```
1  Input: G_SteinerTC = (V_SteinerTC, E_SteinerTC), traffic flow function f
2  Output: p2mp hyperlink function linkset_MP
3  for v ∈ V_SteinerTC do
4  |  linkset_MP := {};
5  end
6  R := V_SteinerTC \ {leaves};
7  while R ≠ {} do
8  |  pick a vertex v ∈ R such that v.depth is maximum;
9  |  linkset_MP(v) := MP-ANT-REPLACE(G_SteinerTC, f, v);
10 |  R := R \ {v};
11 end
12 return linkset_MP;
```

In MP-ANT-REPLACE, the while loop from lines 4 to 32 finds the p2mp antenna with a configuration (direction, beamwidth, radius) covering a maximum number of uncovered vertices at each iteration. The for loop from lines 6 to 17 tries to assign the antenna direction to every vertex u in the available children and then gradually decreases its beamwidth and radius until the constraints (cost, capacity, and interference) are satisfied. The tuning of antenna configuration is presented in the while loop from lines 8 to 12. Let L be the candidate list of the available antenna configurations to be deployed in each iteration, as assigned from lines 13 to 16. After L is computed, from lines 18 to 31, we choose the one that covers the maximum vertices. If no available candidate is found, from lines 18 to 19, we end the replacement at v. Otherwise, we find the candidate with the best coverage in the for loop from lines 22 to 28. Then, we adopt this antenna configuration and remove the vertices that are covered by it. Note that this iteration (loop from lines 4 to 32) ends at line 8 when no configuration satisfies the constraints. Finally, the set of selected antenna configurations is returned as $linkset_{MP}(v)$.

Algorithm 3: MP-ANT-REPLACE

Input: $G_{SteinerTC} = (V_{SteinerTC}, E_{SteinerTC})$, traffic flow function f, candidate vertex v
Output: deployed p2mp antenna set of v, $linkset_{MP}(v) = \{(u_1, BW_1, RAD_{MP1}), \dots\}$
$S := \{u | u \in v.children\}$; $end := false$;
while $S \neq \{\} \wedge end \neq true$ **do**
  $index := 0$;
  for $u \in S$ **do**
    $BW := BWMAX$; $RAD_{MP} := R_{MP}$; $X := \{x | x \in v.children \wedge rng_{MP}((v,x), v, u, BW, RAD_{MP}) = 1\}$;
    while $\left(\frac{cAntenna(MP)}{cAntenna(PP)} < |X| \ \wedge \ (\sum_{x \in X} f((v,x)) \leq U) \ \wedge \ \left(\sum_{e \in (E_{SteinerTC} \setminus \{(v,x) | x \in X\}} rng_{MP}(e, v, u, BW, RAD_{MP}) = 0\right)\right) \neq true \ \wedge \ |X| > 1$ **do**
      decrease BW to the closest vertex $x \in X$ (ignore any vertex at the boundary);
      $X := \{x | x \in v.children \ \wedge \ rng_{MP}((v,x), v, u, BW, RAD_{MP}) = 1\}$;
      decrease RAD_{MP} to the vertex $x \in X$ s.t. $dist(v,x)$ is maximum;
    end
    if $|X| > 1$ **then**
      $L[index] := (u, BW, RAD_{MP})$;
      $index := index + 1$;
    end
  end
  if *index = 0* **then**
    $end := true$;
  else
    $cover_{best} := 0$;
    for $i \leftarrow 0, \dots, index - 1$ **do**
      $cover := \#\{x | x \in v.children \ \wedge \ rng_{MP}((v,x), v, L[i].direction, L[i].BW, L[i].RAD_{MP}) = 1\}$;
      if $cover > cover_{best}$ **then**
        $cover_{best} := cover$;
        select := i;
      end
    end
    $linkset_{MP}(v) := linkset_{MP}(v) \cup L[select]$;
    remove the set of vertices $\{x | x \in v.children \ \wedge \ rng_{MP}((v,x), v, L[select].direction, L[select].BW, L[select].RAD_{MP}) = 1\}$ from S;
  end
end
return $linkset_{MP}(v)$;

3.3.3 Omnindirectional Antenna Deployment

For an omnidirectional hyperlink, we use one omnidirectional antenna to share its capacity, U_{Omni}, with its corresponding subordinate directional antennas. Unlike p2mp antennas, an omnidirectional antenna covers its subordinate directional antennas according to its omnidirectional coverage shaped by R_{Omni}[6]. We assume that the spectrum of the omnidirectional hyperlinks does not overlap with that of the p2p/p2mp links and also that these links are capable of non-line-of-sight propagation; these assumptions are again common; i.e., if we use TVWS omnidirectional hyperlinks with WiFi p2p/p2mp links. The implication of the latter assumption, in particular, is that omnidirectional antenna and their subordinate directional antennas are mounted at a fixed height ($HTOmni$ and $HTOmniSD$) for non-line-of-sight propagation. Hence, the needed tower heights for omnidirectional hyperlinks are considerably shorter.

For Omnidirectional deployment, four constraints must be guaranteed if and only if an omnidirectional hyperlink at a candidate vertex v is going to replace WiFi p2p links associated with the edges $(v, x), x \in X$:

$$
\begin{aligned}
&Cost : cAntenna(Omni) + cAntenna(OmniSD)|X| \\
&\qquad +cTower(\max(HTOmni, h(v))) \\
&\qquad + \textstyle\sum_{u \in V_{OmniSD}} cTower(HTOmniSD) \\
&\qquad < 2 \times cAntenna(PP)|X| + cTower(h(v)) \\
&\qquad + \textstyle\sum_{u \in V_{OmniSD}} cTower(h(u)) \\
&Capacity : \textstyle\sum_{x \in X} f((v, x)) \leq U_{Omni} \\
&Range : dist(v, x) \leq R_{Omni}, (v, x) \in E_{SteinerTC}, x \in X \\
&Interference : \textstyle\sum_{u \in V_{Omni}} rng_{Omni}(u, u.RAD_{Omni}, v, \\
&\qquad\qquad RAD_{Omni}) = 0
\end{aligned}
$$

[6] R_{Omni} varies with the coding schemes (i.e., 64QAM, 16QAM, QPSK) [72]. To support broadband services, we select the one that achieves the best capacity of U_{Omni}.

Since omnidirectional deployment, as formulated, not only affects the antenna cost but also changes the tower heights associated with the hyperlink, the cost constraint guarantees that the total infrastructure cost consisting of link installation and tower deployment must be reduced. The capacity and range constraints ensure that the terminal demands are still satisfied and that the omnidirectional antenna covers the corresponding vertices. The interference constraint guarantees that the deployed omnidirectional antenna at the candidate vertex v does not affect any other omnidirectional hyperlink. The range verification function rng_{Omni} examines if the candidate vertex v with radius RAD_{Omni} overlaps any other omnidirectional antenna at u with its radius $u.RAD_{Omni}$, $u \in V_{Omni}$.

We now state the cost minimization problem of hybrid Omnidirectional deployment as follows:

> ***Input***. *$G_{SteinerTC} = (V_{SteinerTC}, E_{SteinerTC})$, the corresponding traffic flows on edges in $E_{SteinerTC}$, the tower height at vertices in $V_{SteinerTC}$.*
> ***Output***. *A set of omnidirectional antennas with the covered vertices deployed subordinate directional antennas such that the four constraints are satisfied and the cost of towers and wireless links is minimized.*

The solution in Algorithm 4 also considers each candidate vertex in a bottom-up manner and tries to deploy an omnidirectional antenna for its children using Algorithm 5. The algorithm maximizes the cost reduction by maximizing the covered vertices whose tower heights are changed to *HTOmniSD*. Recall that changing a tower height to *HTO mniSD* is profitable due to low cost (i.e., only \$100 for a 10m mast).

In OMNI-ANT-REPLACE, the while loop from lines 5 to 17 finds a maximum set of vertices that we will substitute with subordinate directional antennas. A way to trim the set to satisfy the cost and capacity constraints is shown in lines 6 to 15. First, line 6 verifies if the configuration of the candidate vertex v and radius RAD_{Omni} does

Algorithm 4: OMNI-DEPLOY-ALGO

```
1  Input: G_SteinerTC = (V_SteinerTC, E_SteinerTC), traffic flow function f, height
     function h
2  Output: omnidirectional hyperlink function linkset_Omni
3  V_Omni := {};
4  for v ∈ V_SteinerTC do
5  |  linkset_Omni := {};
6  end
7  R := V_SteinerTC \ {leaves};
8  while R ≠ {} do
9  |  pick a vertex v ∈ R such that v.depth is maximum;
10 |  linkset_Omni(v) := OMNI-ANT-REPLACE(G_SteinerTC, f, h, V_Omni, v);
11 |  if linkset_Omni(v) ≠ {} then
12 |  |  V_Omni := V_Omni ∪ v;
13 |  end
14 |  R := R \ {v};
15 end
16 return linkset_Omni ;
```

not overlap with any other deployed omnidirectional antenna. If there is no overlap, then line 7 examines whether any vertex not in V_{OmniSD} exists. We then remove the one with the maximum flow at line 8. Otherwise, at line 10, we remove the one in V_{OmniSD} with maximum flow. This enhances the chance to satisfy capacity constraints as well as retain more residual capacity. Suppose an overlap exists between the candidate vertex v and any other omnidirectional antenna. From lines 13 to 14, in that case, we remove the vertex in X with maximum distance to v and then retune the radius RAD_{Omni}. After that, we determine from lines 18 to 19 the covered vertices if any set X is found to satisfy the constraints. The covered vertices are then returned as $linkset_{Omni}(v)$.

Algorithm 5: OMNI-ANT-REPLACE

Input: $G_{SteinerTC} = (V_{SteinerTC}, E_{SteinerTC})$, traffic flow function f, height function h, omnidirectional antenna set V_{Omni}, candidate vertex v
Output: deployed subordinate directional antennas of v, $linkset_{Omni}(v) = \{u_1, \ldots, u_{|X|}\}$
$X := \{x | x \in v.children \wedge dist(v, x) \leq R_{Omni}\}$;
$V_{OmniSD} := \{x | x \in X, (v, x)$ is the only induced edge from $x\}$;
while $\big(cAntenna(Omni) + cAntenna(OmniSD)|X| + cTower(\max(HTOmni, h(v))) + \sum_{u \in V_{OmniSD}} cTower(HTOmniSD) < 2 \times cAntenna(PP)|X| + cTower(h(v)) + \sum_{u \in V_{OmniSD}} cTower(h(u)) \ \wedge \ \sum_{x \in X} f((v, x)) \leq U_{Omni}\big) \neq true \ \wedge \ |X| > 0$ **do**
  if $\sum_{u \in V_{Omni}} rng_{Omni}(u, u.RAD_{Omni}, v, RAD_{Omni}) = 0$ **then**
    if $|X \setminus V_{OmniSD}| > 0$ **then**
      remove $x \in X \setminus V_{OmniSD}$ s.t. $f(v, x)$ is maximum, from X;
    else
      remove $x \in X$ s.t. $f(v, x)$ is maximum, from X;
    end
  else
    remove $x \in X$ s.t. $dist(v, x)$ is maximum, from X;
    decrease RAD_{Omni} to the vertex $x \in X$ s.t. $dist(v, x)$ is maximum;
  end
  $V_{OmniSD} := \{x | x \in X, (v, x)$ is the only induced edge from $x\}$;
end
if $|X| > 0$ **then**
  $linkset_{Omni}(v) =$ X;
else
  $linkset_{Omni}(v) = \{\}$;
end
return $linkset_{Omni}(v)$;

3.3.4 Power Assignment and Interference Avoidance

In a homogeneous network (i.e., with only p2p links), the antenna transmit power varies according to the link distance. In order to mitigate the interference between any two p2p links, power selection is possible using, say, the 2P MAC protocol [63] that solves interference issues more efficiently than CSMA/CA for long-distance links. In hybrid networks, in addition, our algorithm for MP / Omnidirectional deployment considers the interference constraint by tuning the transmit radius (and beamwidth of p2mp antennas) to the farthest covered vertex. Hence, the transit power of antennas is assigned during the replacement and causes no additional interference.

3.4 Conclusions

Terrestrial wireless meshes of long-distance links offer an important alternative to address the digital divide in rural areas. Adopting new technologies for high throughput, low latency modalities, such as optical wireless, can further improve their viability. We have focused on the middle-mile network optimization aspect of their sustainable development. We have proposed the first MNO solution that works in polynomial time. Our solution has an approximation ratio of $O(\ln |A| + \frac{|B|}{|A|-1} + \frac{|A|+|B|}{\gamma})$ in tower and antenna cost. We also introduced the SteinerTC problem to ease the MNO problem by determining a network topology covered by minimal tower heights that assure line-of-sight links. Our MNO solution applies to hybrid networks, allowing lower-cost hyperlinks to share capacity and eliminate underutilized capacity.

Chapter 4: QF-MAC: Adaptive, Local Channel Hopping for Interference Avoidance

4.1 Problem Statement

The primary objective of MACs in MCMR mesh networks is to optimize the communication performance of nodes by minimizing the effect of interference within each node's interference region. Doing so impacts various performance metrics of multi-hop flow communications beyond throughput efficiency, reception ratio, latency, and control overhead.

In what follows, we assume that each traffic flow corresponds to a simple path between a source and destination node comprising alternating network node radios and links. Without loss of generality, we let each flow be pinned to a specific radio at the nodes it traverses.

In order to characterize the interference that results from a transmitting radio within its interference region and the efficiency of a channel hopping scheduler, we define a channel efficiency measure. Let $p_{i,ch,t}$ be the probability that radio i at a node sends out a packet over channel ch at time slot t and $\Phi(i)$ denotes the number of internal and external interfering sources within radio i's interference region. (Note that other radios in the same node as i are among the sources that are within its

interference region.) Given sender i and its interference set $\Phi(i)$, the channel efficiency of channel ch within $\Phi(i)$ at time slot t is defined as follows:

$$e_{\Phi(i),ch,t} = \sum_{j\in\Phi(i)} p_{j,ch,t} \prod_{k\in\Phi(i)\setminus\{j\}} (1 - p_{k,ch,t}) \tag{4.1}$$

Let $f(i,t)$ be a channel hopping scheduling function that specifies the channel used at radio i at slot t. Accordingly, the average scheduling efficiency, $\hat{E}$, may be calculated as follows:

$$\hat{E}(f) = \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i\in V} e_{\Phi(i),f(i,t),t} \tag{4.2}$$

where T is the total number of slots, and V is the set of radios in the network. *The channel scheduling problem is to determine a scheduler function f, such that $\hat{E}$ is maximized to minimize the interference.*

All notations related to the channel scheduling problem, and our solution QF-MAC, are listed in Table 4.1.

4.2 QF-MAC Protocol

Towards solving the optimization problem in Eq. 4.2, our design of QF-MAC begins by assigning, for each flow in the network, a channel sequence for each node associated with that flow. Since each flow at a node is pinned to a unique radio, we can equivalently regard its corresponding channel sequence as being associated with the pinned radio. The channel sequence is chosen in coordination with the 1-hop neighbors in the flow, such that intra-flow interference is avoided on an end-to-end basis. Therefore, it allows for concurrent communication along all radios of the same flow. It is moreover chosen such that it deterministically avoids conflicts with the other channel sequences assigned to the other radios in that node (cf. Fig. 4.1).

Table 4.1: Notations for QF-MAC

Symbol	Meaning
U	set of all channels
Δ	interference-to-reliable-transmission ratio
C	number of radios at a node
L	length of a channel hopping sequence
f_x	flow id
$i \in V$	radio id
$(i,j) \in E$	link id
$ch \in U$	channel id
t	time slot number
$\Phi(i)$	set of radios within an interference region
$p_{i,ch,t}$	radio's probability of outgoing packet
$e_{\Phi(i),ch,t}$	channel efficiency in an interference set
$CHS_{Tx/Rx}(f_x,(i,j))$	Tx/Rx channel sequence over a flow link
$S_{\Phi(i)}$	set of channels in all active Tx channel sequences in an interference set
$ag_{\Phi(i),ch,t}$	aggregate channel traffic in an interference set
$G(i,ch)$	radio's goodput efficiency of over a channel

The design of QF-MAC also indirectly deals with internal interference with other nodes, even potentially malicious external interference effects, by adapting the flow sequences as follows. Each node locally updates each of its channel sequences in the presence of varying interference by changing goodput-inefficient channels such that the scheduling efficiency, $\hat{E}$, is optimized over time for the network. Goodput-inefficient channels are identified by leveraging a passive local interference estimation mechanism. Three forms of channel sequence update are performed to improve goodput efficiency–channel augmentation (by adding a channel), channel migration (by substituting a channel), and channel reduction (by dropping a channel).

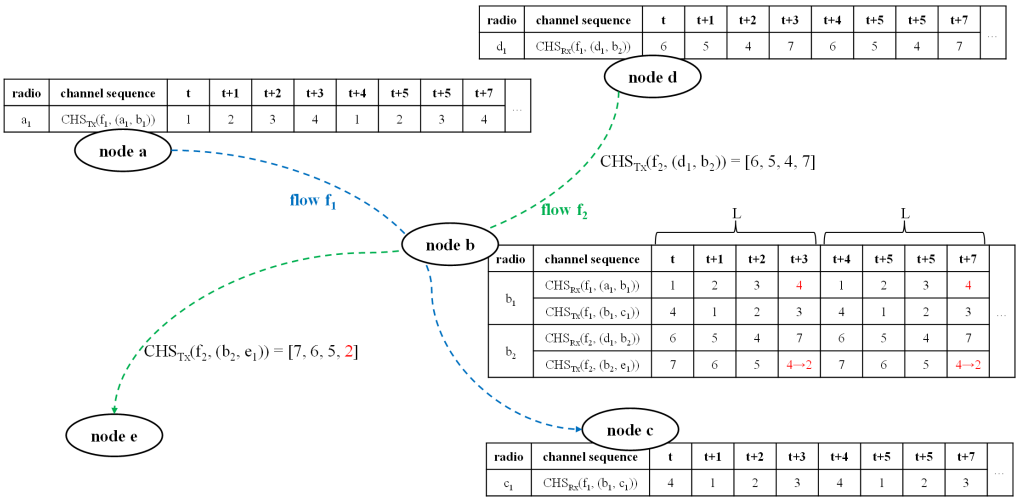


Figure 4.1: Local channel sequences for two flows in a network. The example illustrates how QF-MAC avoids intra-flow and intersecting-flow interference locally. Assume flow f_1 is initiated at radio a_1 with route (a_1, b_1, c_1) to assign Tx and Rx channel sequences in radio a_1, b_1, and c_1. Later flow f_2 is initiated at radio d_1 with route (d_1, b_2, e_1) to send a Tx sequence, $CHS_{Tx}(f_2, (d_1, b_2))$, to radio b_2. Radio b_2 identifies the intersecting-flow conflict between $CHS_{Rx}(f_1, (a_1, b_1))$ and $CHS_{Tx}(f_2, (b_2, e_1))$ and then replaces $ch4$ in $CHS_{Tx}(f_2, (b_2, e_1))$ with a non-conflicted channel, $ch2$, which also avoids intra-flow interference in both f_1 and f_2.

4.2.1 Choosing the Sequence Length for Interference Avoidance

Two factors contribute to characterizing the minimum length of any channel hopping sequence. First, since nodes with C radios can support up to C concurrent transmissions at a node, to avoid self-interference at nodes where concurrent flows intersect, which we refer to as "intersecting-flow" interference, the sequence length L must be at least C. The second factor results from our design wherein we seek to pipeline flows to maximize concurrent transmissions and, thereby, throughput efficiency. To that end, we assume a standard unit-disk interference model [81]. We let Δ denote the ratio of interference distance to the reliable transmission distance, and

R_i and R_t denote the interference radius and reliable transmission radius; $\Delta = \lceil \frac{R_i}{R_t} \rceil$. It follows that to allow concurrent communication over $2\Delta+1$ channels while avoiding intra-flow interference within the interference region, the number of distinct channels in L should be at least $2\Delta+1$.

Hence, the minimum channel hopping sequence length to avoid interference is stated in Eq. 4.3, where U is the set of all channels.

$$|U| \geq L \geq max(2\Delta + 1, C) \tag{4.3}$$

A minimum value of L is then applied to all radios to minimize the control and computation overhead.

4.2.2 Local Assignment of Channel Hopping Sequences

In an accompanying theoretical analysis, relegated to Appendix B, that considers concurrent flows within the interference region of any node, we show that the ratio of collisions/slot of per-flow sequence schedulers (i.e., QF-MAC) is better than that for global sequence schedulers (i.e., TSCH), given $L \leq |U|$. Based on this insight, we design a scheme to assign per-flow channel sequences that notably require only one-hop neighbor coordination.

Assume that a flow has been admitted to the network by choosing its potentially multi-hop routing path and assigning a radio in each of the path nodes to the flow, yielding a route of say $(v_1, ..., v_d)$ with $d-1$ links. Starting with radio v_1, QF-MAC calculates and assigns local channel sequence for the radios in the chosen route; for radio v_r, with $1 < r \leq d$, QF-MAC does so with Algorithm 6. Its channel sequence assignment on v_r minimizes the interference both from all radios associated with the same flow within $\Phi(v_r)$ (intra-flow interference) and from all radios at a node

(intersecting-flow interference). QF-MAC then sends a channel sequence request to the successor v_{r+1} over the control channel and subsequently starts communication over the link (v_r, v_{r+1}) using the channel sequence of v_r. All control messages in QF-MAC are sent via a predefined control channel so that no additional scheduling overhead is involved.

Algorithm 6: CHANNEL-SEQ-ASSIGN at radio v_r

Input: flow id f_x, incoming flow link (v_{r-1}, v_r) with channel sequence $CHS_{Tx}(f_x, (v_{r-1}, v_r))$, outgoing flow link (v_r, v_{r+1})
Output: channel sequence $CHS_{Tx}(f_x, (v_r, v_{r+1}))$
if v_r = *src*(f_x) **then**
    $CHS_{Tx}(f_x, (v_r, v_{r+1}))$:= L non-repeating random channels selected from U;
    schedule send of $CHS_{Tx}(f_x, (v_r, v_{r+1}))$ to v_{r+1};
else
    $CHS_{Rx}(f_x, (v_{r-1}, v_r)) := CHS_{Tx}(f_x, (v_{r-1}, v_r))$;
    if v_r != *dest*(f_x) **then**
        $CHS_{Tx}(f_x, (v_r, v_{r+1})) := RIGHT{-}SHIFT(CHS_{Tx}(f_x, (v_{r-1}, v_r)), 1)$;
    end
    if $(\exists\ l, f_y$ *s.t.* $CHS_{Rx}(f_x, (v_{r-1}, v_r))[l] = CHS_{Rx}(f_y, (v_q, v_r))[l])$ **then**
        schedule send of reject of $CHS_{Tx}(f_x, (v_{r-1}, v_r))$ to v_{r-1};
    else
        if $v_r \neq$ *dest*(f_x) $\wedge$ $(\exists\ l, f_y$ *s.t.* $CHS_{Tx}(f_x, (v_r, v_{r+1}))[l] = CHS_{Rx}(f_y, (v_q, v_r))[l])$ **then**
            replace $CHS_{Tx}(f_x, (v_r, v_{r+1}))[l]$ with a channel without conflict;
        end
        if $v_r \neq$ *dest*(f_x) $\wedge$ $(\exists\ l, f_y$ *s.t.* $CHS_{Tx}(f_x, (v_r, v_{r+1}))[l] = CHS_{Tx}(f_y, (v_r, v_s))[l])$ **then**
            replace $CHS_{Tx}(f_x, (v_r, v_{r+1}))[l]$ with a channel without conflict;
        end
        if $(\exists\ l, f_y$ *s.t.* $CHS_{Rx}(f_x, (v_{r-1}, v_r))[l] = CHS_{Tx}(f_y, v_r, v_s)[l])$ **then**
            replace $CHS_{Tx}(f_y, (v_r, v_s))[l]$ with a channel without conflict;
            schedule send of $CHS_{Tx}(f_y, (v_r, v_s))$ to v_s;
        end
    end
    schedule to send $CHS_{Tx}(f_x, (v_r, v_{r+1}))$ to v_{r+1};
end

Algorithm 7: RIGHT-SHIFT(CHS, offset)

```
1 for l = 0..L − 1 do
2 |  CHS'[(l + offset)%L] := CHS[l];
3 end
4 return CHS';
```

I. Intra-flow Interference Avoidance. When a flow f_x is initiated given a route $(v_1, ..., v_r, ..., v_d)$, the source radio v_1 first randomly selects a channel sequence $CHS_{Tx}(f_x, (v_1, v_2))$ consisting of L non-repeating random channels for link (v_1, v_2) and then sends the channel sequence to the next hop, v_2. This is shown in Lines 3 to 5 of Algorithm 6. Upon receiving the channel sequence from v_{r-1}, as shown in Line 7, v_r sets up the scheduling for data reception over (v_{r-1}, v_r). If v_r has an outgoing link for f_x, as shown in Lines 8 to 9, it then adopts by default the channel sequence for the transmissions on link (v_r, v_{r+1}) right shifted by one position, according to Algorithm 7, so as to avoid intra-flow interference between the two links (v_{r-1}, v_r) and (v_r, v_{r+1}). Along the flow path, the intra-flow interference is thus successively avoided by repeating this assignment until the destination v_d is reached.[7]

As will be described in Section 4.2.2.II, a radio v_r may need to change its channel sequence from the default one to accommodate channel sequence conflicts from the other flows through its node. To avoid intra-flow interference in this case as well, the channel sequence request sent to v_{r+1} includes not just the channel sequence of v_r but also the sequence information of $v_r's$ $\Delta-1$-hop ancestors[8]. Accordingly, v_{r+1}

[7]For paths with a round shape, intra-flow interference may result if v_r and v_{r+L} are within $\Phi(v_r)$ and $(OF_{v_r} - OF_{v_{r+L}})$ mod $L = 0$. We handle this case implicitly via the inter-flow interference handling Channel Adaptation mechanism, described in Section 4.2.3.

[8]QF-MAC allows each radio to change its Tx channel sequence locally to avoid intersecting-flow interference. The update is nontrivial since $v_r's$ $\Delta-1$-hop ancestors can use different channel sequences.

avoids intra-flow interference using the potentially different channel sequences being used within its interference region in the flow. We observe in passing that a feasible assignment for avoiding intra-flow and intersecting-flow interference exists as long as the assumption that L satisfies Eq. 4.3 holds.

II. Intersecting-flow Interference Avoidance. Since each node is equipped with C full-duplex radios, it can perform up to C concurrent Tx+Rx communications. Each radio v_r associated with flow f_x, has its incoming Rx channel sequence, $CHS_{Rx}(f_x, (v_{r-1}, v_r))$, and its outgoing Tx channel sequence, $CHS_{Tx}(f_x, (v_r, v_{r+1}))$. To avoid interference with the flows at other active radios at the same node, QF-MAC checks if any channel in $CHS_{Rx}(f_x, (v_{r-1}, v_r))$ or $CHS_{Tx}(f_x, (v_r, v_{r+1}))$ conflicts with some other radio's Tx/Rx channel sequences. In order to save control overhead, QF-MAC allows v_r to locally change only its Tx channel sequence of the link (v_r, v_{r+1}) to avoid interference; the net result is that the channel sequence scheduling of a path continues to be completed in a downstream direction.

The change is accomplished as follows: As shown in Lines 11–24 in Algorithm 6, four types of conflicts are considered: (Rx_{f_x}, Rx_{f_y}), (Rx_{f_x}, Tx_{f_y}), (Tx_{f_x}, Rx_{f_y}), and (Tx_{f_x}, Tx_{f_y}) conflicts, wherein the left and right terms of a tuple stand respectively for the newly scheduled sequences of flow f_x and the existing sequence from other radios associated with flow f_y. Lines 11–12 specify the handling of the (Rx_{f_x}, Rx_{f_y}) conflict case: Since v_r is not allowed to modify the Rx channel sequence of (v_{r-1}, v_r), it sends a reject with its channel sequence scheduling information to v_{r-1} so that v_{r-1} can reschedule a channel sequence for link (v_{r-1}, v_r). In the (Tx_{f_x}, Rx_{f_y}) conflict case, as shown in Lines 14–15, and the (Tx_{f_x}, Tx_{f_y}) conflict case, as shown in Lines 17–18, the conflicted channel in $CHS_{Tx}(f_x, (v_r, v_{r+1}))$ is replaced directly to exclude

the collision. Note that the replacement should exclude all the conflicted channels used in both the other radios at the node itself and the $\Delta-1$-hop ancestors of flow f_x. As shown in Lines 20–22, the case of (Rx_{f_x}, Tx_{f_y}) is resolved by changing the Tx channel sequence of f_y and then sending the updated channel sequence request to the corresponding next hop, v_s.

Note that an update of $CHS_{Tx}(f_y, (v_r, v_s))$ may cause intra-flow interference at the radios associated with f_y. This case is handled implicitly via the inter-flow interference handling in Channel Adaptation (cf. Section 4.2.3).

4.2.3 Channel Adaptation

Unlike intra-flow and intersecting-flow interference, the variation of inter-flow and external interference across node locations and time is not controlled by QF-MAC. QF-MAC does, however, adapt to these interference dynamics to optimize channel efficiency over time. We recall that Eq. 4.1 states that channel efficiency $e_{\Phi(i),ch,t}$ is optimized when $\sum_{v\in\Phi(i)} p_{v,ch,t} = 1$[9]. Moreover, we recall that [46] shows that $\sum_{v\in\Phi(i)} p_{v,ch,t}$ can be approximated by a local estimation of $p_{i,ch,t} + e^{I(ch,i,t)}$, where $I(ch,i,t)$ is an interference estimator of the probability that some interferers $j \in \Phi(i) \setminus \{i\}$ transmit on channel ch at slot t. And furthermore, IEEE 802.11 and IEEE 802.15.4 utilize the evaluation of clear channel access (CCA) in the physical layer to detect the level of interference, $I(ch,i,t)$. All of these observations make it feasible in QF-MAC to have each radio i locally and efficiently estimate channel efficiency $e_{\Phi(i),ch,t}$, by measuring incoming traffic, $p_{i,ch,t}$, and the interfering traffic on channel

[9]Namely, in channel hopping schedulers, only one radio transmits a packet over ch at t within $\Phi(i)$.

ch. Notably, this estimation is achieved without introducing any extra communication overhead.

By adopting the use of packet acknowledgments for transmissions, QF-MAC also calculates the goodput efficiency, $G(i, ch)$, in terms of the ratio of ACKed packets to transmitted packets at radio i on channel ch, and thereby characterizes the effect of external interference. In turn, it uses the estimates of the channel efficiency and the goodput efficiency to mitigate inter-flow and external interference, as follows.

Inter-flow and External Interference Avoidance

Let $S_{\Phi(i)}$ be the set of channels used in all active Tx channel sequences from radios in $\Phi(i)$, $|U| \geq |S_{\Phi(i)}| \geq L$[10]. Also, let $ag_{i,ch,t} = \sum_{j \in \Phi(i)} p_{j,ch,t}$ denote the aggregate traffic in channel ch in the interference set of radio i at slot t, It follows that $\sum_{ch \in S} ag_{\Phi(i),ch,t} > |S_{\Phi(i)}|$ implies that in $\Phi(i)$ the aggregate traffic load from all channels in $S_{\Phi(i)}$ exceeds the achievable capacity. QF-MAC improves the transmission performance at i by introducing a channel out of $S_{\Phi(i)}$ to replace the one with the largest $ag_{\Phi(i),ch,t}$ in all active Tx channel sequences at the node. On the other hand, $\sum_{ch \in S} ag_{\Phi(i),ch,t} \leq |S_{\Phi(i)}|$ and a poor value of $G(i, ch)$ reveal an increased impact from external interference on a particular channel. QF-MAC selects a new channel for radio i from $S_{\Phi(i)}$ to replace the channel with the lowest $G(i, ch)$ among all of its Tx channel sequences. Algorithm 8 specifies the rules of channel adaptation with respect to three cases: channel augmentation, channel migration, and channel reduction.

Channel augmentation, as shown in Lines 1–6, is triggered whenever the aggregate traffic load over all channels in $S_{\Phi(i)}$ is larger than the achievable capacity. In Line 2, radio i selects for replacement a channel ch_{old} whose traffic within its interference

[10] $S_{\Phi(i)}$ can be estimated through monitoring CCA signals over all channels at radio i.

Algorithm 8: CHANNEL-ADAPT(i)

1 **if** $\sum_{ch \in S} ag_{\Phi(i),ch,t} > |S_{\Phi(i)}|$ **then**
2 $\quad ch_{old} = \text{argmax}_{ch \in S_{\Phi(i)}} ag_{\Phi(i),ch,t}$;
3 $\quad X := U \setminus S_{\Phi(i)}$;
4 $\quad$ select a random $ch_{new} \in X$;
5 $\quad$ replace ch_{old} by ch_{new} in $\{CHS_{Tx}(i,j) \mid ch_{old} \in CHS_{Tx}(i,j)\}$;
6 $\quad$ send updated $CHS_{Tx}(i,j)$ to j;
7 **else**
8 $\quad$ **if** $(\exists\ ch \in S_{\Phi(i)} : G(i,ch) < \theta_1)$ **then**
9 $\quad\quad ch_{old} = \text{argmin}_{ch \in S_{\Phi(i)}} G(i,ch)$;
10 $\quad\quad X :=$ all non-conflicting channels in $S_{\Phi(i)}$;
11 $\quad$ **else if** $\sum_{ch \in S_{\Phi(i)}} ag_{\Phi(i),ch,t} < \theta_2 |S_{\Phi(i)}|$ **then**
12 $\quad\quad ch_{old} = \text{argmin}_{ch \in S_{\Phi(i)}} ag_{\Phi(i),ch,t}$;
13 $\quad\quad X :=$ all non-conflicting channels in $S_{\Phi(i)}$;
14 $\quad$ select $ch_{new} \in X$ with probability $\frac{ag^{-1}_{\Phi(i),ch_{new},t}}{\sum_{ch \in X} ag^{-1}_{\Phi(i),ch,t}}$;
15 $\quad$ replace ch_{old} by ch_{new} in $\{CHS_{Tx}(i,j) \mid ch_{old} \in CHS_{Tx}(i,j)\}$;
16 $\quad$ send updated $CHS_{Tx}(i,j)$ to j;
17 **end**

region is the largest. Note that $ag_{\Phi(i),ch_{old},t}$ is always greater than 1, which implies a high level of inter-flow interference and hence the likelihood that the substitution of ch_{old} at radio i will improve the performance at radio i. In Lines 3–4, a new channel from $U \setminus S_{\Phi(i)}$ is introduced to increase the number of channels used for concurrent transmission in $\Phi(i)$ to reduce the internal interference. In Lines 5–6, the newly selected channel replaces ch_{old} in all Tx channel sequences used in radio i, and then it notifies the corresponding neighbor about the update of the channel sequence.

Channel migration, as shown in Lines 8–10, swaps out the most goodput-inefficient channel from Tx sequences. Note that Line 8 captures the case with short-term interference dynamics–i.e., $ag_{\Phi(i),ch,t} \leq |S_{\Phi(i)}|$–but where at least one channel has goodput below the threshold of θ_1. In Lines 9–10, radio i replaces the channel, ch_{old},

with minimal goodput efficiency with a new channel from $S_{\Phi(i)}$, selected in Line 14, that does not conflict any other channels in radio i's Tx and Rx channel sequences at t. The probability of selecting a channel is inversely proportional to its aggregate traffic. Instead of deterministic selection, we adopt a probabilistic rule that eschews the situation where nodes with similar estimated states will simultaneously swap to the same channel. Then the update of Tx channel sequences, in Lines 15–16, is sent to the receiver j of link (i, j).

Channel reduction, as shown in Lines 11–13, shrinks the size of $S_{\Phi(i)}$ if the average channel efficiency is below a threshold θ_2. Note that channel efficiency is maximized when $ag_{\Phi(i),ch,t} = 1$. Radio i reduces the size of $S_{\Phi(i)}$ to be close to L when the average data traffic within its interference region is low. This further reduces the overhead needed to accurately monitor channel states in $S_{\Phi(i)}$.

4.2.4 Compatibility with Routing Protocols

QF-MAC works not only for path-oriented routing (i.e., on-demand routing) but also with table-driven routing protocols. In table-driven routing, the routing path of each packet may vary as the routing metrics change over time. When the path changes, QF-MAC deals with resulting changes as follows: If radio i's next forwarder changes from j to k in its routing table, whenever the first packet arrives at i after this change, radio i sends a channel sequence request to k and a channel sequence cancel to j, respectively, with only one-hop communication. Moreover, for the intra-flow interference from nodes in the flow at more than one hop distance, it effectively treats that as inter-flow interference: it spontaneously performs channel adaptation

based on the estimation of the aggregate outgoing load and the interfering traffic in an interference region.

4.3 Evaluation

4.3.1 Configuration Space of Simulated Networks

Our validation of QF-MAC uses the ns3 simulator, which we extended in several ways: to support MCMR communication at each node, to support jammer nodes, and to emulate our real-code implementation of the MAC. (The extension to support emulation is based on an integration of ns-3 with the Direct Code Execution framework [4]). The validation compares the performance of QF-MAC with that of TSCH and CSMA/CA, in terms of goodput, end-to-end latency, control overhead, and packet reception ratio. It does so for environments with and without external interference and node mobility. To calibrate the impact of leveraging local interference estimation and channel adaptation in tolerating adversarial jamming, we simulate two versions of QuickFire MAC: one with channel adaptation (QF-MAC-A) and the other without (QF-MAC).

All MACs are tested with point-to-point flows whose routes are, in all cases, set up and maintained via a common reactive path-oriented routing protocol that finds a minimum interference path over the data plane for each flow.

In the MCMR meshes we simulate, each node is equipped with four radios ($C = 4$) that share a fixed capacity of 8 Mbps. For QF-MAC and TSCH, each radio operates in one of 8 channels, each with 1Mbps capacity. One channel is dedicated to sharing control messages (for routing and MAC); the remaining seven channels are used for data communication, i.e., $|U| = 7$. For CSMA, each node operates with one dedicated

channel of 1 Mbps for the control plane and two channels of 3.5 Mbps for data plane communication; when serving more than two flows, each radio uses round-robin scheduling for the flows. We note that we have used a modest number of channels and radios per node in part because that represents the state-of-the-art but also because it conveniently lets us study the ability of these MACs to handle interference. For the data plane, we apply a time-slotted MAC communication with a 10 ms slot for QF-MAC and TSCH and a 3 ms slot for CSMA. Every slot is long enough to transmit a 1000-byte data packet and receive an acknowledgment between a pair of nodes within some transmission radius.

The length of the channel sequence is set to its minimal value, 4, for QF-MAC and the maximal value (without repeated channel), $|U| = 7$, for TSCH. These assignments of L satisfy Eq. 4.3 while allowing for reduced control overhead for QF-MAC and the best opportunity for TSCH to avoid interference.

Concerning interference, each radio is configured with the same transmission and interference radius of 1km ($\Delta = 1$) followed by a unit-disk interference model [81]. And with respect to simulation scenarios with external interference, a single adversarial jammer with a +3 dB above the nominal transmission power and a 100% effective jamming range of 1.32km is placed in the center of the network layout. Outside of this radius, jamming contributes to node-local RF noise conditions and could still cause blocking interference depending on the SNR margin of the potential victim link, just as any other simulated RF emitter would. The jammer continually generates jamming signals to reduce the network data capacity by 2Mbps (i.e., from 7Mbps to 5Mbps) within its interference region, as follows: For QF-MAC and TSCH, it jams the two channels with the highest aggregate traffic. For CSMA, it jams over

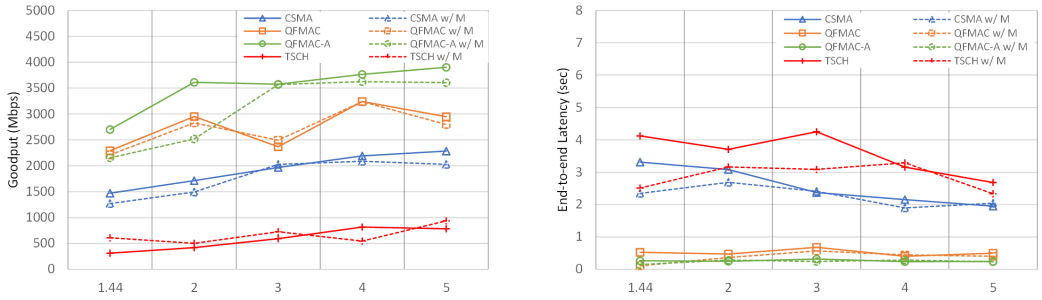
a sub-band within one of the two 3.5 Mbps channels to reduce the available capacity by 2 Mbps.

The space of experiments we conducted is 4-dimensional, namely: (i) network size in $\{64, 125, 216\}$, (ii) network density[11] in $\{\sqrt[3]{3}, 2, 3, 4, 5\}$, (iii) number of concurrent 3 Mbit flows over in $\{1, 3, 7, 10\}$, and (iv) mobility in $\{0, 10\}$ m/s followed by Gauss-Markov model. With respect to these four dimensions, we simulated networks with nodes distributed over a rectangular region using a uniform random distribution placement model. Each of our experiment configurations consists of four trials of 1 minute. All flows concurrently arrive after 5 seconds from the start of each trial. We note that each flow generates one packet at each slot. Each marked point in the lines in Figs. 4.2 and 4.4 represents an average result taken over the different numbers of flows.

4.3.2 Performance in Static and Mobile Networks

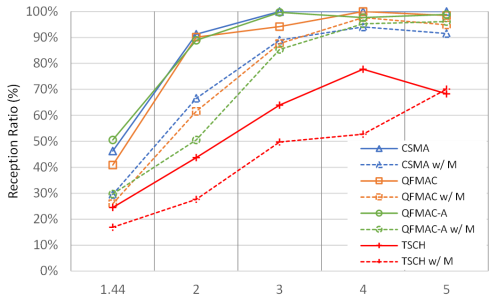
We begin by comparing goodput performance: Figs. 4.2(a) and 4.3 show that QF-MAC versions substantially outperform TSCH and CSMA over all configurations of density, size, and mobility. CSMA, even operating with two wideband data channels of 3.5Mbps, does not yield high goodput efficiency because of frequent transmission delays of concurrent transmissions, which can be inferred from Fig. 4.2(b). TSCH has the lowest (and sub-1Mbps) goodput across almost all configurations, largely due to frequent retransmissions during concurrent communications; recall that with a global shared channel sequence, there are continuous collisions over time when $\Phi(i) > L$. This indicates that TSCH cannot sustain reliable communication in networks when there

[11]Density is defined as the average number of nodes per R_t^2 area, with R_t denoting the transmission range

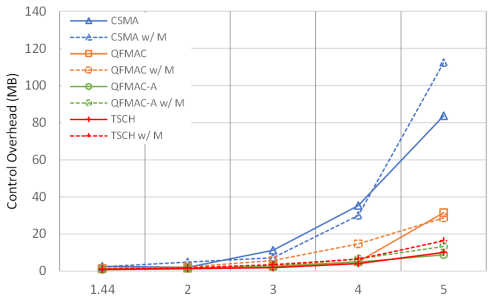


(a) Average Goodput

(b) Average Latency

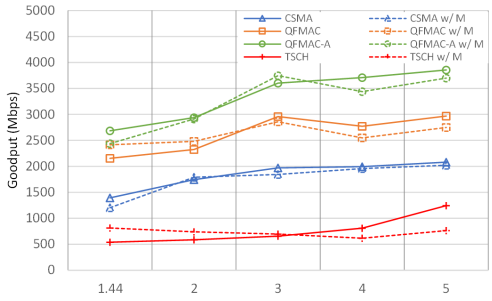


(c) Average Packet Reception Ratio

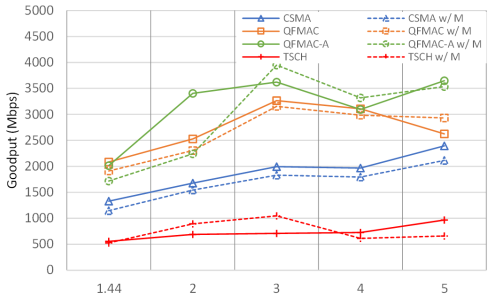


(d) Average Overhead

Figure 4.2: Density versus different performance metrics for QuickFire MAC versus CSMA and TSCH at network size of 125. The solid and dashed (denoted with "w/ M") lines respectively denote network scenarios that are static and mobile.



(a) 64 nodes



(b) 216 nodes

Figure 4.3: Density versus Goodput for QuickFire MAC versus CSMA and TSCH at network sizes of 64 and 216.

are many concurrent flows. Notably, QF-MAC with channel adaptation (QF-MAC-A) offers a remarkable improvement over the version without adaptation (QF-MAC) for both static and mobile network scenarios. With respect to network density, from $\sqrt[3]{3}$ to 2, QF-MAC and QF-MAC-A deliver more goodput as the density increases, but the goodput tends to decrease as the density increases 3 or 4 over different network sizes because of increasing impacts of inter-flow interference. We observe that mobility does not always hurt goodput: on occasion, it offers alternative paths that better avoid inter-flow interference, in which case goodput can be higher than in the static case.

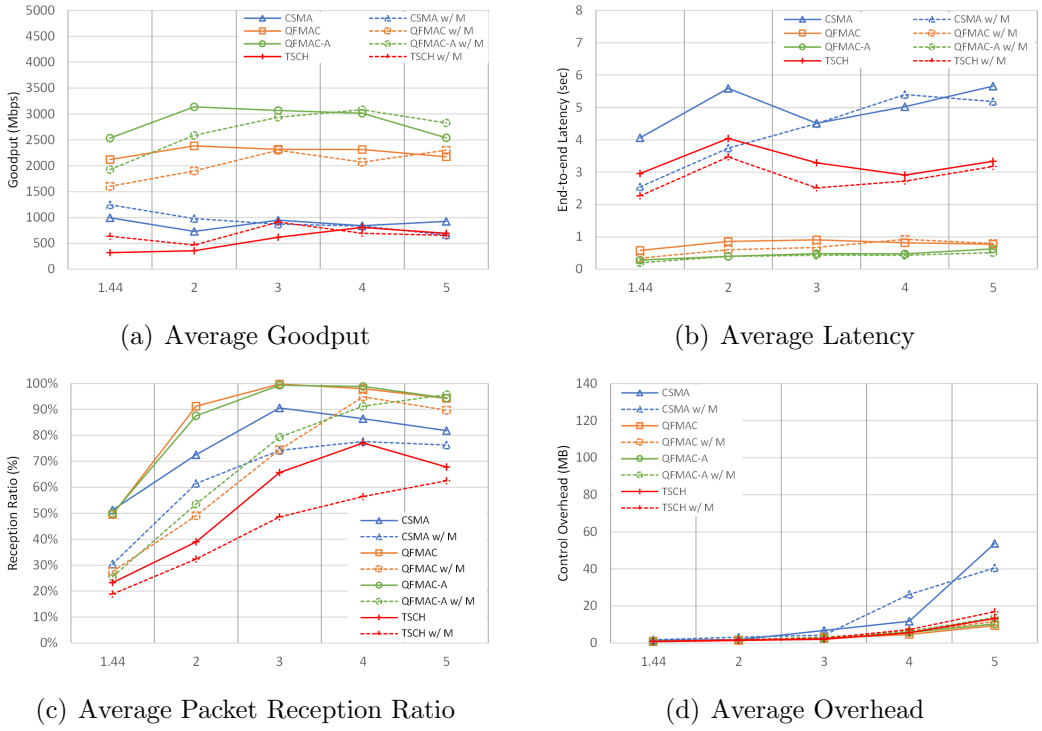
With respect to end-to-end latency comparisons, Figs. 4.2(b) likewise show that CSMA and TSCH have lower performance than QF-MAC and QF-MAC-A, again due to the frequent transmission delay of CSMA and the larger number of transmissions retries of TSCH. QF-MAC-A likewise better mitigates the impact of inter-flow interference by swapping out a transmission channel with the worst performance. In packet reception ratio comparisons, Figs. 4.2(c) demonstrates that the use of 1Mbps channels in QF-MAC versions achieves similar reliability compared to the use of 3.5Mbps channels in CSMA. This is due to the effective avoidance of intra-flow interference and the reduction of inter-flow interference in QF-MAC. Again, the low reception ratio of TSCH is caused by frequent collisions among concurrent flows. We observe a sizeable gap in packet reception ratio between the static and mobile cases: this is because mobility renders the routing state stale and thus undermines the reliability of path routing.

With respect to control overhead comparisons of the MAC and routing stacks, Fig. 4.2(d) shows that even though CSMA incurs no MAC communication overhead

to configure transmission channels, CSMA yields the largest control overhead across all densities, both with and without mobility. QF-MAC-A performs comparably to TSCH. We note that, in dense networks, arriving flows wait longer to be served as the completion time increases in the flows currently served. This is because each node serves at most four flows in a first-come-first-served fashion. As a result, unserved flows tend to perform a higher number of route exploration rounds in CSMA and TSCH, which also contributes to a higher end-to-end latency than that QF-MAC/QF-MAC-A, as seen in Fig. 4.2(b). In contrast, QF-MAC-A efficiently leverages the trade-off between the added control overhead for channel adaptation and the goodput efficiency to achieve substantially better goodput and end-to-end latency.

4.3.3 Performance with respect to Adversarial Jamming

Recall that the adversarial jammer reduces the capacity of the network within its interference region by 2Mbps, in QF-MAC and TSCH, jamming two channels that have the highest aggregate traffic and, in CSMA, one of the two data channels. Figs. 4.4(a) and 4.5 show that QF-MAC versions effectively eschew the jammed channels and substantially outperform CSMA and TSCH across varying densities, sizes, and mobility speeds. Observe that QF-MAC and QF-MAC-A have respectively ∼20% and ∼25% goodput reduction, whereas CSMA has a ∼52% drop of goodput. This implies that both the use of local channel sequences and the use of channel adaptation help mitigate the loss of achievable capacity in the presence of adversarial jamming. TSCH, with a 10% goodput reduction, again shows poor performance of sub-1Mbps because of its lack of adaptivity to internal and external interference.



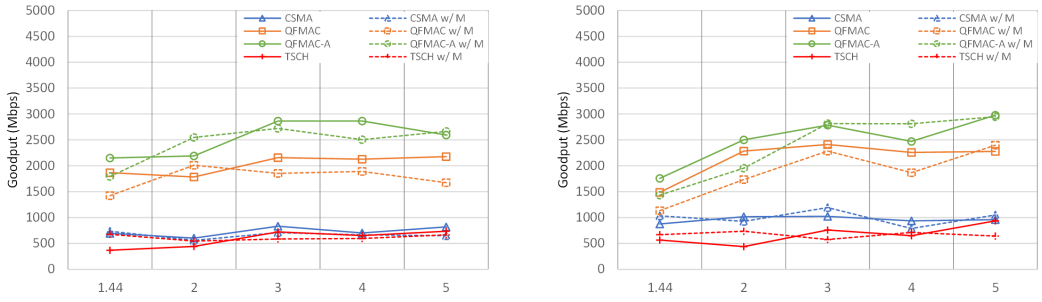
(a) Average Goodput

(b) Average Latency

(c) Average Packet Reception Ratio

(d) Average Overhead

Figure 4.4: Density versus different performance metrics for QuickFire MAC versus CSMA and TSCH at network size of 125 in the presence of adversarial jamming.



(a) 64 nodes

(b) 216 nodes

Figure 4.5: Density versus Goodput for QuickFire MAC versus CSMA and TSCH at different network sizes of 64 and 216 with adversarial jamming.

In end-to-end latency comparisons, Fig. 4.4(b) shows that CSMA and TSCH have a much higher latency than QF-MAC and QF-MAC-A in the presence of jamming. In contrast, QF-MAC versions achieve a sub-1 second latency over all configurations due to the efficient channel utilization over the channels free from jamming.

Regarding reception ratio comparisons, Figs. 4.2(c) and 4.4(c) show that QF-MAC and QF-MAC-A are still able to sustain reception ratio under adversarial jamming. CSMA suffers an obvious reliability decrease of over 10% once the density crosses 3, with and without mobility. Since the effect of inter-flow interference from concurrent flows yields numerous retransmissions in TSCH, it again achieves a poor reception radio in the presence of external jamming. With respect to control overhead comparisons, Fig. 4.4(d) shows essentially the same overhead comparison as in the case without adversarial jamming.[12]

4.4 Conclusion

By leveraging a local channel hopping schedule for multi-hop communication, QF-MAC adapts across diverse network scenarios (e.g., of density, concurrency, mobility, etc.) and is robust to internal and external interference in multi-radio networks. In contrast, TSCH-based protocols, which rely on a link offset schedule to stagger a global channel hopping sequence, are prone to more collisions or otherwise incur goodput-latency performance penalties. Similarly, CSMA-based protocols, albeit more robust to collisions than TSCH, underperform in goodput-latency and thus have

[12]Surprisingly, the control overhead of QF-MAC-A and CSMA decreases when jamming is introduced, especially once the density increases to 3 or more, but this is due to a subtle artifact of the routing protocol, which chooses the minimum interference path in the data plane and essentially finds paths that circumvent the jammed region of the network. A detailed explanation of this reduction is outside the scope of this paper, as this pattern is not seen for other routing protocols.

relatively high control overhead as well. By allowing nodes to leverage local interference estimation to further adapt their channels, QF-MAC achieves a gain in channel efficiency with low control overhead, even in the presence of high interference dynamics. QF-MAC works with on-demand routing over single or multiple paths and is also compatible with table-driven routing protocols.

Chapter 5: QF-Geo: Capacity Aware Geographic Routing using Bounded Regions

5.1 Analysis of Path Stretch and Boundedness of Search Region

This section describes Monte Carlo simulations that quantify the path stretch. The objective of the simulation is to demonstrate the correlation between path stretch and the network parameters in uniform random networks. Furthermore, by analyzing the simulation results, we determine a probability bound ($\mathbb{P} = 99\%$) on path stretch and thus justify exploring routing paths within an ellipse predicted by a model with only a few inputs of network parameters.

5.1.1 Simulating the Path Stretch

Let d_e denote the Euclidean distance between two endpoints and let d_{sp} denote the length of the shortest path between the endpoints. Define unitless versions of these distances as $\delta := d_e/R$ and $s := d_{sp}/R$, where R is the transmission radius. Further, define the path stretch as the ratio s/δ and denote it by ζ. Clearly, $1 \leq \zeta$ because the path length can never be less than the Euclidean distance. And if $\delta \leq R$, then $\zeta = 1$ because there is a direct link (i.e., a one-hop path) between the endpoints. Note that if two nodes are not connected, then $s = \infty$ and $\zeta = \infty$. Since the probability of at

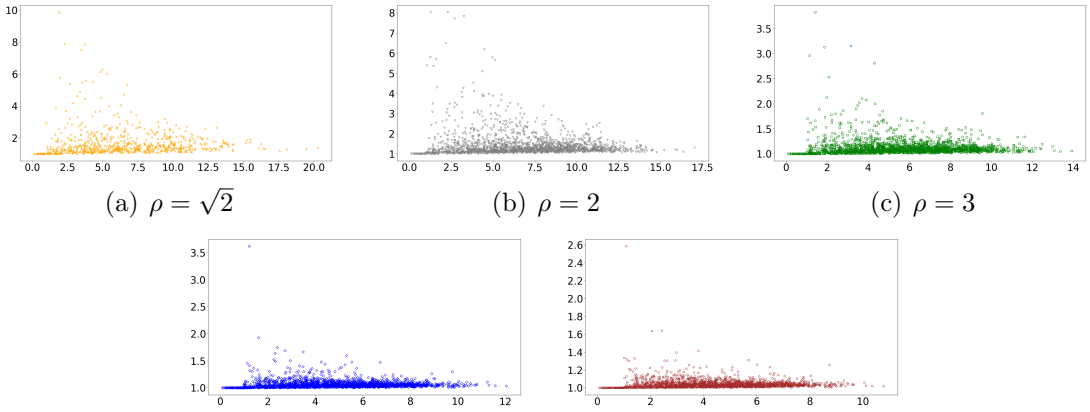
least some network nodes being disconnected is non-zero, in the statistics, we only count those networks for which there exists at least one route between the given two endpoints.

To generate samples from a 2D distribution of (δ, ζ), we construct a network with node locations of uniform random distribution and links determined by geometric proximity. Let n denote the number of nodes, and ρ denote the network density with the definition of the average number of nodes per R^2 area. Then all nodes are distributed in a square with the side $w = \sqrt{\frac{n \times R^2}{\rho}}$. Our analysis is based on 2000 networks per configuration with $n = 343$ and $\rho \in \{\sqrt{2}, 2, 3, 4, 5\}$. We select a fixed pair of endpoints (v_s, v_d) in each example and compute the shortest paths between these two endpoints accordingly. Since the examples without connectivity between the selected endpoints are filtered out, the number of networks per configuration we count in the analysis may vary and be less than 2000.

5.1.2 Distribution of Path Stretch

Fig. 5.1 plots the path stretch across different network densities. The subfigures show that the variance becomes more significant as δ decreases. Also, given a value of δ, the corresponding average ζ reduces as the network density decreases. Notice for short connections that, while there is a substantial chance of less than a 1% net path stretch, there is also a substantial chance of path stretch that ranges from up to 10x to up to 2.6x as the network density ranges over $\{\sqrt{2}, 2, 3, 4, 5\}$. These observations imply that, with high probability, there is a tight bound on path stretch ζ, and thereby a shortest path exists in a limited network region that is determined by ρ and δ. We are therefore motivated to validate if bounded search regions can be associated that,

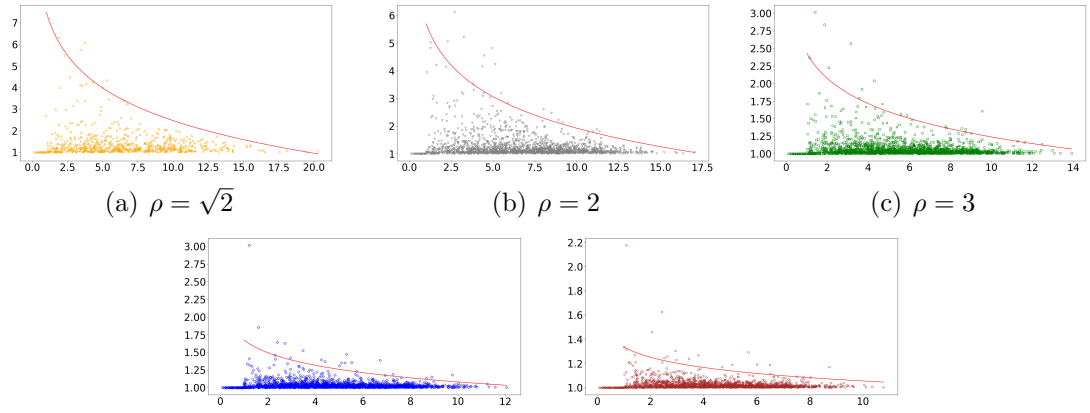
with high probability, contain a shortest path and for which it is feasible to predict the size of the regions in terms of the ρ and δ parameters.



(a) $\rho = \sqrt{2}$ (b) $\rho = 2$ (c) $\rho = 3$

(d) $\rho = 4$ (e) $\rho = 5$

Figure 5.1: Scatter plot of path stretch (δ, ζ) for endpoints (v_s, v_d) in 2000 random networks per configuration with $n = 343$ and density $\rho \in \{\sqrt{2}, 2, 3, 4, 5\}$.



(a) $\rho = \sqrt{2}$ (b) $\rho = 2$ (c) $\rho = 3$

(d) $\rho = 4$ (e) $\rho = 5$

Figure 5.2: Scatter plot of ellipse factor (δ, ℓ_{con}) for endpoints (v_s, v_d) of the same network graphs as in Fig. 5.1. The red curves denote the 99th percentile bound of ℓ_{con} derived from quantile regression.

5.1.3 Simulating the Bounded Search Region

Given a graph and the two endpoints (v_s, v_d) in it as foci, the ellipse factor ℓ_{con} can be defined to shape the ellipse whose major axis is $\ell_{con} \times \overline{v_s v_d}$. For a path pt, we define V_{pt} as the set of nodes on path pt. Then the ellipse factor in determining the minimum-sized ellipse to include pt can be obtained as follows:

$$\ell_{con} = \frac{max(\overline{v_s v_r} + \overline{v_r v_d})}{\overline{v_s v_d}}, v_r \in V_{pt} \tag{5.1}$$

To simulate the distribution of the bounded search region across different ρ and δ, we calculate ℓ_{con} from the shortest path dataset in Fig. 5.1.

5.1.4 The Ellipse Factor for Achieving Connectivity

Fig. 5.2 shows the distribution of ℓ_{con} respectively for densities ranging over $\rho \in \{\sqrt{2}, 2, 3, 4, 5\}$. The red curve in each subfigure represents the 99th percentile bound of the ellipse factor; the bound is computed using quantile regression [43] in a logarithmic scale with a model of $w_1 \ln \delta + w_0$ over the weights w_1 and w_0. The curves show bounds on the ℓ_{con} such that the shortest path can, with high probability, be explored in a limited region determined by ρ and δ. Note that the ellipse with the factor of ℓ_{con} may include paths pt longer than the shortest path sp, whose corresponding ellipse factor ℓ_p is less than ℓ_{con}, but searching in those smaller ellipses would not be desirable as it would likely yield a suboptimal route. The search region should be the minimum subgraph that includes the shortest path.

5.1.5 The Model for Prediction of ℓ_{con}

Our objective is to predict the minimum search region of the shortest path to route packets according to the network density and source-destination distance. To

that end, we propose the following model to calculate ℓ_{con} given ρ and δ:

$$\ell_{con}(\rho, \delta) = \begin{cases} 1, \ if \ \delta \leq 1 \\ max(1 + \frac{\alpha \ln \delta + \beta}{\rho^{\gamma}}, \ell_{min}), \ otherwise, \end{cases} \tag{5.2}$$

where α, β, and γ are coefficients that determine the size of an ellipse in accordance with the network parameters (i.e., ρ and δ). The lower bounds of ℓ_{con}, 1 and ℓ_{min}, are respectively provided for $\delta \leq 1$ and $\delta > 1$.

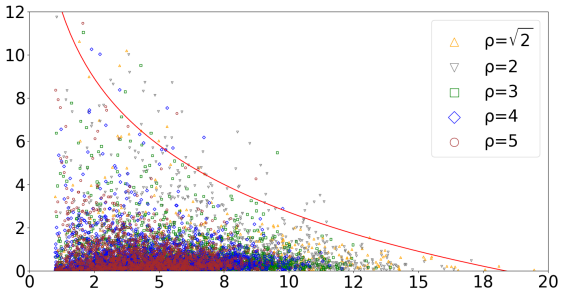


Figure 5.3: Scatter plot of normalized ellipse factor $(\delta, \hat{\ell}_{con})$. The red curve denotes the 99th percentile bound of $\hat{\ell}_{con}$ derived from quantile regression.

The parameters of the model are calculated as follows. First, we determine the value of γ such that normalized values of ℓ_{con} ($\hat{\ell}_{con} = (\ell_{con} - 1) * \rho^{\gamma}$) in the training dataset have a similar distribution over the y-axis across different densities. Thus, we choose $\gamma = 2$; the corresponding distribution of $(\delta, \hat{\ell}_{con})$ is plotted in Fig. 5.3. Next, we compute α and β using quantile regression at the 99th percentile using a standard logarithmic model, $\hat{\ell}_{con} = \alpha \ln \delta + \beta$. The red curve in Fig. 5.3 represents the 99th percentile bound of normalized ellipse factor $\hat{\ell}_{con}$. Table 5.1 shows for each $\rho \in \{\sqrt{2}, 2, 3, 4, 5\}$ the percentage of the ellipse factors ℓ_{con}, which are predicted by Eq. 5.2, that are below the red curves in Fig. 5.2; in each case, the percentage is close to the targeted 99%.

Table 5.1: Percentage of values of ℓ_{con}, predicted by Eq. 5.2 with $\alpha = -4.4732$, $\beta = 13.0715$, $\gamma = 2$, $\ell_{min} = 1.05$, that are below the 99th percentile bound for $\rho \in \{\sqrt{2}, 2, 3, 4, 5\}$ in Fig. 5.2.

$\rho = \sqrt{2}$	$\rho = 2$	$\rho = 3$	$\rho = 4$	$\rho = 5$
98.53%	97.27%	99.59%	99.48%	98.69%

5.2 The *QF-Geo* Protocol

5.2.1 Overview of *QF-Geo*

QF-Geo performs per-packet routing between two endpoints (v_s, v_d) within an ellipse computed by a function of network density, source-destination distance, and current traffic. To predict the minimum size of the ellipse, which offers requirements for connectivity (i.e., a subgraph including the shortest path) and capacity (i.e., a subgraph including sufficient residual capacity), *QF-Geo* respectively calculates ℓ_{con} and ℓ_{cap}. The ℓ_{con}, predicted by the model in Eq. 5.2, is the ellipse factor that determines a search region in which nodes v_r included in a shortest path, $\overline{v_s v_r} + \overline{v_r v_d} \leq \ell_{con} \overline{v_s v_d}$, can be found with probability $\mathbb{P}$. In the presence of concurrent communication and external interference, we calculate another ellipse factor ℓ_{cap}, $\ell_{cap} \geq \ell_{con}$, to heuristically approximate a minimum area that provides for sufficiently many forwarders whose residual capacity Θ satisfies the capacity requirement C.

QF-Geo builds on the inherent local search in geographic routing. Recall that in geographic routing, a suboptimal path is found by greedily forwarding packets to the neighbor closest to the destination in a high-density network when geometrically direct routes exist. However, in a foam-like network, backtracking may be required

when encountering holes without prior knowledge. Thus, in order to ensure goodput efficiency and reliable packet delivery, *QF-Geo* applies a depth-first search scheme to traverse the subgraph within a designated elliptic area. Each packet keeps information on visited nodes to distinguish unvisited neighbors and maintains a stack of the explored path to determine the forwarder when backtracking. To further improve performance, *QF-Geo* prioritizes forwarding relay selection to avoid relays whose capacity is below a threshold [39]; it also maintains a per-flow memory of the last forwarder so as to eschew forwarding to nodes that will backtrack and thus save the routing search time. To accommodate potential network mobility, it also uses an ϵ-greedy approach to leverage exploitation and exploration activities to adapt to the topology change. With a probability of ϵ, *QF-Geo* explores a new route so as to adapt to network changes.

5.2.2 The Ellipse Factor for Achieving Capacity

For capacity-aware forwarding, each source node v maintains an estimate of the density $\rho'(\leq \rho)$ of a subgraph of the network that includes only nodes whose residual capacity exceeds a capacity threshold C_{min}. To that end, it periodically estimates its residual capacity $\Theta(v)$ by observing the outgoing data traffic and interference traffic.[13] $\Theta(v)$ is used not only to predict ℓ_{cap} but also as the metric used to prune forwarding candidates. Also, by collecting $\Theta(u)$ from node u within v's neighborhood, v locally builds a distribution model to estimate the probability distribution of residual capacity in a region centered around v. Thus, given a capacity threshold C_{min}, v is able to

[13]The residual capacity $\Theta(v)$ can be approximated by a local estimation of $max(1-(D_v+e^{I(v)}),0)$ [46], where D_v is the probability that node v sends out a packet and $I(v)$ is an interference estimator of the probability that some interferers transmit within v's interference range.

predict the percentage of nodes with a residual capacity greater than C_{min}, denoted by $\phi \in [0, 1]$.

Using the effective density $\rho' = \phi\rho$ and the source-destination distance, ℓ_{cap} is calculated by the model in Eq. 5.2:

$$\ell_{cap}(\rho, \delta, \phi) = \ell_{con}(\phi\rho, \delta). \tag{5.3}$$

Note that underestimation of ℓ_{cap} may result in forwarding packets to bottlenecks and thus cause transmission failure. For a given packet that exceeds a retransmission limit, *QF-Geo*, therefore, allows any relay v_r holding this packet to recalculate ℓ_{cap} used for routing between v_r and v_d.

5.2.3 The *QF-Geo* Algorithm

Algorithm 9 shows *QF-Geo*'s handling of a flow that arrives at node v. In Lines 1 and 2, it first predicts the ellipse factor ℓ through the function L that calculates ℓ_{cap} from Eq. 5.3 at either the source node or the relay hits the maximum retransmission times. It then forwards packet p within the ellipse with factor ℓ. The actions for forwarding packet p at v are specified in Algorithm 10. Relevant notations are listed in Table 5.2.

Algorithm 9: *QF-Geo*(p, v)

```
1 if v = p.src ∨ RETX(p, v) > RETX_max then
2 |   p.ℓ := L(p.src, p.dst, ρ');
3 end
4 FORWARD(p, v, ℓ);
```

Table 5.2: Notations for QuickFire

Symbol	Meaning
p	packet id
ℓ	ellipse factor
src	source node id
dst	destination node id
fl	flow id
ρ'	effective network density
$\Theta(v)$	estimated residual capacity of node v
$C(fl)$	capacity requirement of flow fl
$RETX(p, v)$	retransmission times of packet p at node v
$RETX_{max}$	packet retransmission limit at a forwarder
$BL(p, v)$	set of neighbors backtracking to v for packet p
$q(fl, v)$	last forwarding target at v in flow fl
$NE(p, v, \ell)$	set of v's neighbors within the forwarding ellipse of p
$Dist(v, u)$	Euclidean distance between nodes v and u
$Parent(p, v)$	parent of node v in packet p's traversed path
$V(p)$	set of visited nodes for packet p
rnd	random value $\in [0, 1]$

The first line in Algorithm 10 filters the candidate forwarders from node v's neighbors, which are located within the bounded region of p. Note that since *QF-Geo* applies a depth-first search, these candidates should be unvisited and excluded from the nodes that are backtracking to v. In addition to satisfying the capacity requirement, the forwarding prioritizes the candidates with sufficient residual capacity $\Theta(u)$ for flow $p.fl$. Lines 2 to 10 specify the ϵ-greedy approach, which forwards packets when the set of candidates is not empty. Packet p is sent immediately to the last forwarder of flow $p.fl$ with a probability of $1 - \epsilon$. Otherwise, v forwards p to the node closest to p's destination. Lines 11 to 16 handle the case where there is no qualified candidate filtered in Line 1; node v still tries to find a neighbor with the best routing progress

Algorithm 10: FORWARD(p, v, ℓ)

```
1  S := {u | u ∈ NE(p, v, ℓ) ∧ u ∉ V(p) ∧ u ∉ BL(p, v) ∧ Θ(u) ≥ C(p.fl)};
2  if S ≠ {} then
3  |  if rnd < 1 − ε ∧ q(p.fl, v) ∈ S then
4  |  |  forward p to u := q(p.fl, v);
5  |  else
6  |  |  forward p to u := argmin_{x∈S} Dist(x, p.dst);
7  |  |  q(p.fl, v) := u;
8  |  end
9  |  Parent(p, u) := v;
10 |  V(p) := V(p) ∪ {v};
11 else if S = {} ∧ {u | u ∈ NE(p, v, ℓ) ∧ u ∉ V(p)} ≠ {} then
12 |  S := {u | u ∈ NE(p, v, ℓ) ∧ j ∉ V(p)};
13 |  forward p to u := argmin_{x∈S} Dist(x, p.dst);
14 |  q(p.fl, v) := u;
15 |  Parent(p, u) := v;
16 |  V(p) := V(p) ∪ {v};
17 else
18 |  backtrack to u := Parent(p, v);
19 |  BL(p, u) := BL(p, u) ∪ v;
20 end
```

to $p.dst$. Finally, in Lines 17 to 19, if all of v's neighbors within the forwarding region are already visited, v must backtrack packet p to its parent u. In that case, u marks v in $BL(p, u)$ so that v will not be the prioritized forwarder in p's associated flow.

5.3 Evaluation

5.3.1 Configuration Space of Network Emulation

We use the same network configurations described in Section 4.3. The validation compares the performance of *QF-Geo* with that of greedy forwarding (GF) and maximum capacity routing (MCR). GF is the geographic routing that greedily forwards packets to the nodes closest to the destination. MCR is a reactive protocol

that collects regional network states within an elliptic region predicted by Eq. 5.2 to find the maximum capacity route for the entire flow transmission. To also calibrate the impact of bounded forwarding, we simulate two versions of *QF-Geo*: one using a bounded region (*QF-Geo*) and the other without a bounded region (*QF-Geo*-A). All routing protocols are tested with point-to-point flows and the MAC communication over CSMA.

Each of our experiment configurations consists of four trials of 30 seconds. Each trial starts with no assumption of each node's neighborhood or network state. To obtain the states of the neighborhood (e.g., coordinates and regional capacity Θ), each node periodically broadcasts a Hello message over the control channel every 200 ms. All flows concurrently arrive after 5 seconds from the start of each trial. We note that each flow generates one packet at each slot. Once a flow arrives, *QF-Geo* and GF immediately start forwarding packets at the corresponding source nodes. In MCR, the arrival of flows at the source triggers route exploration to retrieve the state of the subgraph within a bounded ellipse, and then the source calculates the maximum capacity path for routing. At the end of each trial, the destination of a flow updates the received bits, flow time, and end-to-end latency for calculating the performance metrics.

In terms of the performance metrics, we compare the goodput, packet reception ratio, and average latency of different protocols. For the network configurations with concurrent flow transmission, we calculate an average result taken over the different numbers of flows with packets received at their destination. The goodput is defined as $\frac{total\ received\ bits}{average\ flow\ time}$, where the flow time of a single flow is the interval between the first sent packet at the source and the last received packet at the destination during the

simulation time. Note that a high goodput can be resulted not only from a flow with reliable path(s) that transmit all packets but also from a flow with path(s) that deliver only a few packets in a short time. We thus propose the metric of goodput efficiency, which takes into account both the goodput and the reception ratio to evaluate the communication efficiency and reliability of a routing protocol in diverse scenarios. Goodput efficiency is defined as follows:

$$goodput\ efficiency = \frac{goodput \times reception\ ratio}{bandwidth \times \#flows} \in [0, 1], \tag{5.4}$$

where bandwidth represents the bit rate over the entire radio spectrum. Perfect goodput efficiency is achievable in flows with a one-hop path that is free from interference and collision.

5.3.2 Performance in Static and Mobile Networks

Figs. 5.4 and 5.5 show that *QF-Geo* works well over all configurations of density, size, and mobility. Note that the average source-destination distance reduces as the network size decreases from 125 to 64. This change reduces the average length of routing paths and may then yield lower latency.

In static network scenarios (Fig. 5.4), although GF high goodput efficiency in fog-like networks ($\rho = 4, 5$) since its direct forwarding enjoys a high reception ratio and low latency from a dense node distribution. Yet, it underperforms in goodput efficiency and reception ratios in foam-like (i.e., minimally connected such as with $\rho = \sqrt{2}$) networks; this low performance is due to failures in its handling of holes. It has good latencies overall, given its bias towards direct paths, but this comes at the expense of reliability. In contrast, MCR receives high goodput efficiency at density $\sqrt{2}$ but low goodput efficiency at densities of 2, 3, 4, and 5, because its average flow

(a) Goodput Efficiency, Size 125

(b) Goodput Efficiency, Size 64

(c) Packet Reception Ratio, Size 125

(d) Packet Reception Ratio, Size 64

(e) Latency (sec), Size 125

(f) Latency (sec), Size 64

Figure 5.4: Density versus different performance metrics for *QF-Geo* versus GF and MCR at networks with 125 and 64 static nodes. We use different fill patterns of columns to represent the three types of connected networks: large grids denote foam-like networks ($\rho = 1.4, 2$), diamond grids denote connected networks between foam-like and fog-like ones ($\rho = 3$), and small grids denote fog-like networks ($\rho = 4, 5$). Each point-to-point flow is 3Mbit. Goodput efficiency and packet reception ratio are typically improved by *QF-Geo*; its latency is competitive, especially if its improved reception ratio in the presence of interference is accounted for.

(a) Goodput Efficiency, Size 125

(b) Goodput Efficiency, Size 64

(c) Packet Reception Ratio, Size 125

(d) Packet Reception Ratio, Size 64

(e) Latency (sec), Size 125

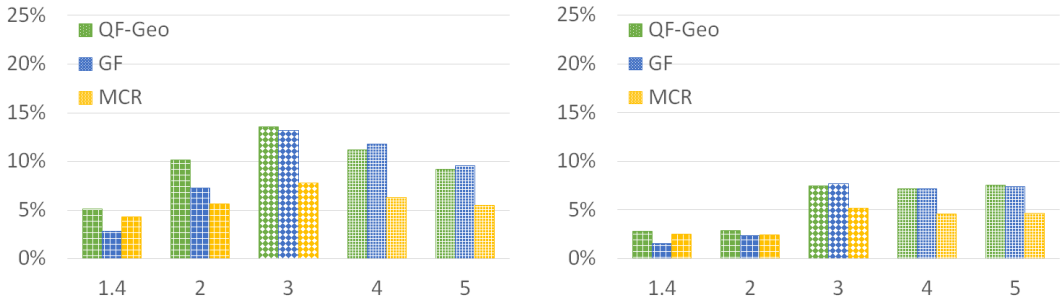
(f) Latency (sec), Size 64

Figure 5.5: Density versus different performance metrics for *QF-Geo* versus GF and MCR at networks with 125 and 64 mobile nodes. Goodput efficiency, reliability, and latency are typically improved by *QF-Geo*.

time includes a non-negligible portion of exploration time that significantly reduces the goodput efficiency. Its reliability tends to be high because it searches for the

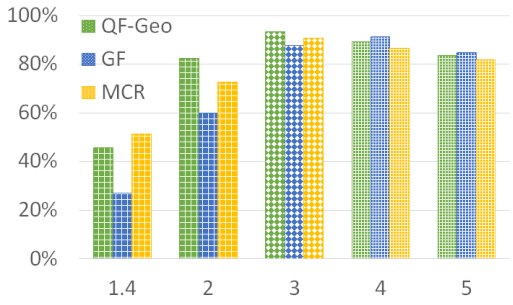
best paths in its region of exploration. The results of MCR imply that a snapshot-based protocol always incurs overhead for network state collection, even if that is within a bounded region. The resulting exploration time is insufficiently amortized over the data transmission time, especially in flows with a small amount of data, and the latencies are consistently high as well. Compared with both GF and MCR, *QF-Geo* achieves significantly better goodput efficiency and reliability in foam-like networks; and even in fog-like networks, it does consistently better than MCR and is competitive with GF. Overall, the results show that *QF-Geo* effectively generalizes to the scenarios over various densities in the static case.

In mobile network scenarios (Fig. 5.5), across all configurations, *QF-Geo* substantially outperforms GF and MCR in terms of goodput efficiency and packet reception ratio and at least significantly outperform MCR in terms of latency. Importantly, there is no obvious gap in the performance between the static and mobile networks in *QF-Geo*. This is because its forwarder prioritization and memory of the last forwarder sustain the reliability of routing paths and the communication efficiency over moving nodes. Since a packet can be rapidly forwarded to the destination along the pre-explored route, *QF-Geo* efficiently reduces the chance of backtracking packets by avoiding frequent exploration of new routes. On the other hand, the path reliability in GF and MCR is undermined as nodes become mobile. Unlike *QF-Geo*'s filtering out of underperforming relays, GF is prone to forward packets over long links, which are likely broken. Furthermore, MCR may suffer from the staleness of the network state, causing it to choose unreliable routing paths.

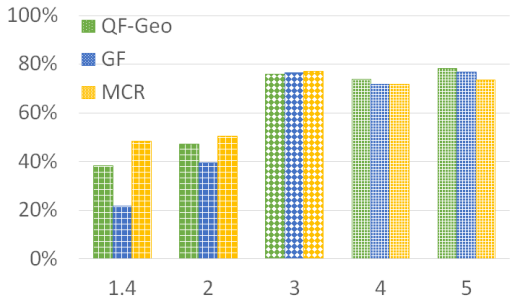


(a) Goodput Efficiency, Size 125

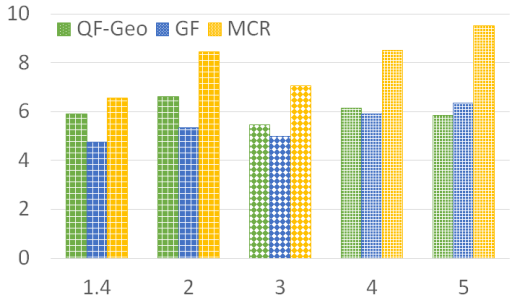
(b) Goodput Efficiency, Size 64



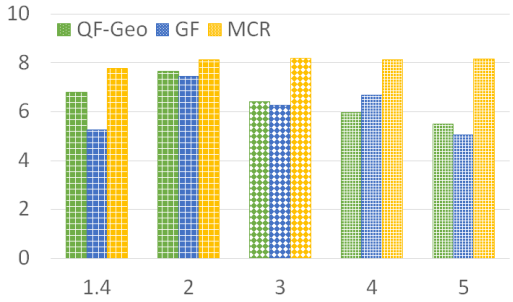
(c) Packet Reception Ratio, Size 125



(d) Packet Reception Ratio, Size 64



(e) Latency (sec), Size 125



(f) Latency (sec), Size 64

Figure 5.6: Density versus different performance metrics for *QF-Geo* versus GF and MCR at networks with 125 and 64 static nodes and an additional jammer located at the center of network.

Performance with respect to Jamming

Figs. 5.6 and 5.7 show that *QF-Geo* effectively eschews the jammed region to achieve high goodput efficiency and packet reliability across varying densities, sizes,

(a) Goodput Efficiency, Size 125

(b) Goodput Efficiency, Size 64

(c) Packet Reception Ratio, Size 125

(d) Packet Reception Ratio, Size 64

(e) Latency (sec), Size 125

(f) Latency (sec), Size 64

Figure 5.7: Density versus different performance metrics for *QF-Geo* versus GF and MCR at networks with 125 static nodes and an additional jammer located at the center of network.

and mobility speeds. We note that as the network size decreases from 125 to 64, the number of flows interfered with by the jamming region increases, and then the performance is likely to degrade.

In static network scenarios, *QF-Geo* consistently outperforms in terms of goodput efficiency and packet reliability in foam-like networks and remains competitive in fog-like networks. Compared to *QF-Geo* and GF, MCR does not achieve the highest reception ratio in high-density networks because its calculation of bounded region is not aware of the current jamming traffic. A failure to extend the ellipse area (and thus increase the space diversity) likely yields more interference as the degree of concurrent communication and the external jamming traffic increase. Therefore, *QF-Geo* achieves comparably good performance metrics in all density configurations due to the adaptation of ellipse size to both density and network capacity.

In mobile network scenarios, *QF-Geo* clearly outperforms GF and MCR in terms of goodput efficiency and reception ratio across all configurations; its latency is competitive with GF and better than that of MCR. *QF-Geo*'s performance in mobile cases is sometimes even higher than that of *QF-Geo* in static cases; this is because mobility may offer additional chances to route packets away from jammed regions.

5.3.3 Impact of Bounded Forwarding on Improved Performance

Table 5.3: The improvement of performance metrics in *QF-Geo* compared to *QF-Geo*-A.

Network Scenario		Goodput Efficiency		Reception Ratio		Latency	
		w/o jammer	w/ jammer	w/o jammer	w/ jammer	w/o jammer	w/ jammer
Static	foam-like	+0.93%	+4.40%	+0.78%	+1.89%	-0.04%	-1.92%
	fog-like	0%	+0.14%	0%	-0.05%	0%	-0.68%
Mobile	foam-like	-1.12%	+0.49%	-0.40%	-1.09%	-1.50%	-6.96%
	fog-like	0%	+0.05%	0%	-0.01%	0%	-0.13%

In this section, we compare two versions of *QF-Geo* , *QF-Geo* (with a bounded region) and *QF-Geo*-A (without a bounded region), to show the effect of forwarding packets within an ellipse whp including the shortest path. The results show on average bounded forwarding achieves comparable performance in terms of goodput efficiency and reception ratio but can further reduce the end-to-end latency.

Table 5.3 compares the difference in performance metrics across configurations of mobility and density between *QF-Geo* and *QF-Geo*-A. The green and red fonts respectively denote positive and negative percentage differences in the metrics between the two versions. On average, the difference in goodput efficiency and reception ratio is very slight, indicating that bounding is sufficient for capturing the efficiency. We note that in the scenarios of fog-like networks without jamming traffic, bounded forwarding yields an identical performance to the unbounded case because the routing paths with high capacity are geometrically direct and can be greedily found by both *QF-Geo* and *QF-Geo*-A. Also, the cases with increased reception ratio in *QF-Geo* imply that bounded forwarding may achieve more packet delivery ratio in a limited experiment period. However, it may also reduce the reception ratio in low-density networks since, with low probability, a routing path does not exist in a bounded region for some flows.

The results also show that on average and typically, *QF-Geo* has an improved latency across all configurations. In foam-like networks, *QF-Geo* shows a reduction of latency up to 1.50% in cases without jamming traffic and 6.96% in cases with jamming traffic by using a bounded forwarding scheme. We note that the cases with the most reduction of latency happen in the configurations of mobility and foam-like networks. This implies that using bounded forwarding can effectively reduce the

length of routing paths in sparse networks, and the gain becomes more significant while frequently dealing with broken links in mobile networks.

5.4 Conclusions

A probabilistic framework allows for a tighter bound on the region of exploration than a deterministic framework while only rarely failing to find a shortest path.

We analyzed the geometric region of forwarding consistent with high goodput efficiency and reliability and found that across the full range of densities, this region is well approximated by an ellipse parameterized by the source-destination distance and the density. For foam-like scenarios, the region of sufficient exploration is a "fat" ellipse; and for fog-like networks, the region is a "skiny" elongated ellipse. This approach contrasts with most existing routing algorithms, which make implicit assumptions that limit their effectiveness to either fog-like or foam-like network densities.

We presented a geographic routing protocol based on bounded exploration over these elliptical regions that are robust to network dynamics with densities, including mobility, external interference, and changes in traffic patterns. Via network emulation, we experimentally validated that substantial improvements in goodput efficiency, reliability, and end-to-end latency are achieved at scales of up to several hundreds of nodes.

Chapter 6: Learning from A Single Graph for Near-Shortest Path Routing

6.1 Problem Formulation for Generalized Routing

Consider the class $\mathbb{G}$ of all graphs $G = (V, E)$ whose nodes are uniformly randomly distributed over a 2-dimensional geometric space, that is either an Euclidean space or a hyperbolic space. Each node $v \in V$ knows its global coordinates. For any pair of nodes $v, u \in V$, edge $(v, u) \in E$ holds if and only if the distance between v and u is at most the communication radius R.

For the case of G in a 2-dimensional Euclidean plane, we let R be a user-defined constant. Let ρ denote the network density, where network density is defined to be the average number of nodes per R^2 area, and n the number of nodes in V. It follows that all nodes in V are distributed in a square whose side is of length $\sqrt{\frac{n \times R^2}{\rho}}$.

For the case of G in a 2-dimensional hyperbolic plane, all nodes in V are distributed in a disk of radius R. Each node v thus has hyperbolic polar coordinates (r_v, θ_v) with $r_v \in [0, R]$ and $\theta_v \in [0, 2\pi)$. Let δ be the average node degree. Let n and α denote the number of nodes in V and the negative curvature, respectively. All nodes have their radial coordinates sampled according to the probability distribution $p(r) = \alpha \frac{\sinh(\alpha r)}{\cosh(\alpha R) - 1}$ and angular coordinates sampled uniformly at random from the

interval $[0, 2\pi)$. It is well known that such uniform random graphs in the hyperbolic plane yield a power-law distribution for the node degrees [7].

6.1.1 All-Pairs Near-Shortest Path Problem

The objective of APNSP routing problem is to locally compute for all node pairs of any graph $G \in \mathbb{G}$ their near-shortest path. Here, near-shortest path is defined as one whose length is within a user-specified factor (≥ 1) of the shortest path length.

Formally, let $d_e(O, D)$ denote the distance between two endpoints O and D, and $d_{sp}(O, D)$ denote the length of the shortest path between these endpoints. Further, let $\zeta(O, D)$ denote the path stretch of the endpoints, i.e., the ratio $\frac{d_{sp}(O,D)}{d_e(O,D)}$.

The APNSP Problem. Learn a routing policy π such that, for any graph $G = (V, E) \in \mathbb{G}$ and any origin-destination pair (O, D) where $O, D \in V$, $\pi(O, D, v) = u$ finds v's next forwarder u and in turn yields the routing path $p(O, D)$ with path length $d_p(O, D)$ that with high probability is a near-shortest path. In other words, π optimizes the accuracy of $p(O, D)$ as follows:

$$\max \quad Accuracy_{G,\pi} = \frac{\sum_{O,D \in V} \eta(O, D)}{|V|^2}, \tag{6.1}$$

$$s.t. \quad \eta(O, D) = \begin{cases} 1, & if \ \frac{d_p(O,D)}{d_{sp}(O,D)} \leq \zeta(O, D)(1 + \epsilon) \\ 0, & otherwise \end{cases} \tag{6.2}$$

Note that the user-specified factor for APNSP is $\zeta(O, D)(1+\epsilon)$, where $\epsilon \geq 0$. The use of path stretch $\zeta(O, D)$ in Equation 6.2 is to normalize the margin for shortest paths prediction (given ϵ is constant) across networks with different densities since the variance of path stretch becomes more significant as the network density ρ decreases[14].

[14]The result in [19] shows that, in sparse graphs (e.g., ρ=1.4, 2), the path stretch can vary in $[1.0, 10.0]$ since some of the shortest paths are prone to be found around the holes. However, the path stretch in dense graphs (e.g., ρ=4, 5) is likely to vary only within the range of $[1.0, 3.5]$. Thus, we exploit $\zeta(O, D)$ to mitigate the gap between sparse and dense networks for APNSP prediction.

6.1.2 MDP Formulation for the APNSP Problem

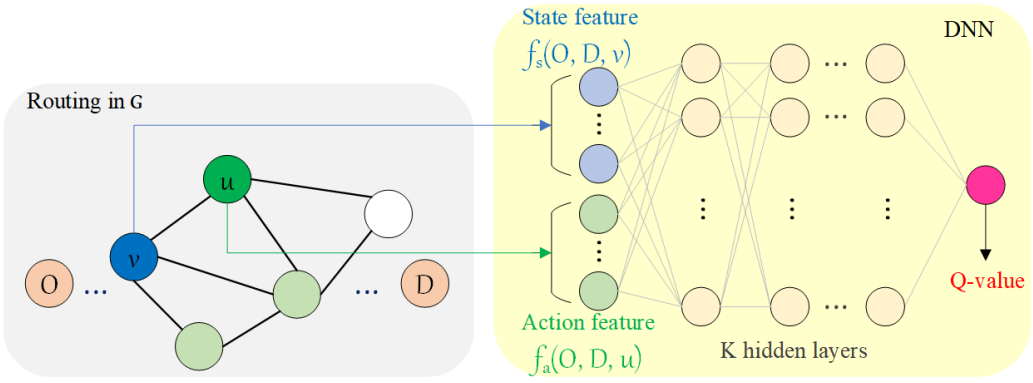


Figure 6.1: Schema for solution using DNN to predict Q-values for selecting the routing forwarder.

To solve APNSP, we first formulate it as a Markov decision process (MDP) problem that learns to choose actions that maximize the expected future reward. In general, an MDP consists of a set of states S, a set of actions $A(s)$ for any state $s \in S$, an instantaneous reward $r(s, a)$, indicating the immediate reward for taking action $a \in A(s)$ at state s, and a state transition function $P(s'|s, a)$ which characterizes the probability that the system transits to the next state $s' \in S$ after taking action a at the current state $s \in S$. To simplify the routing behavior in the problem, the state transition is assumed to be deterministic. Specifically, each state s represents the features of a node holding a packet associated with an origin-destination pair (O, D), and an action $a \in A(s)$ indicates the routing behavior to forward the packet from the node to one of its neighbors. Given the current state s and an action

$a \in A(s)$ which selects one neighbor as the next forwarder, the next state s' is determined as the features of the selected neighbor such that the probability $P(s'|s,a)$ is always one. The tuple (s,a,r,s') is observed whenever a packet is forwarded. Note that we assume that every node implicitly knows the coordinates of the origin and the destination. For convenience, (O,D) will be hidden in the following presentation of state s, which will denote only the information of a routing node v, $s := v$ rather than $s := (O,D,v)$.

In addition, we define the Q-value to specify the cumulative future reward from state s and taking action a: $Q(s_t = s, a_t = a) = \sum_{i=t}^{L} \gamma^{i-t} r(s_i, a_i)$, where $\gamma, 0 \leq \gamma \leq 1$, is the discount factor, and L is the hop distance of the shortest path from the current node to the destination. When $\gamma = 0$, the instantaneous reward is considered exclusively, whereas the future rewards are treated as equally important as the instantaneous reward in the Q-value if $\gamma = 1$. In the APNSP problem, we define the instantaneous reward $r(s,a)$ as the negative length of the corresponding edge (s,s'), and set $\gamma = 1$. Therefore, the optimal Q-value, $Q^*(s,a)$, is equal to the cumulative negative length of the shortest path from s to the destination.

For solving APNSP, we seek to learn the optimal Q-value through a data-driven approach with DNNs. As depicted in Figure 6.1, each state s and action a will be embedded into a set of input features denoted by $f_s(s)$ and $f_a(a)$, respectively. A DNN will be learned to approximate the optimal Q-value given the input features, based on which a near-shortest path routing policy can be obtained by choosing actions with largest Q-values.

6.1.3 Design of Input Features

To learn a routing protocol that generalizes across graphs with different scales, densities, and topologies, the input features of the DNN should be designed to be independent of global network configurations, including the identity of nodes and of packets. Recall that each node knows its own coordinates and the coordinates of the origin and the destination.

For feature selection, we first apply the decision tree and symbolic regression [13] to filter the crucial features from a set of candidate features widely used in geographic routing protocols: distance from v to destination D, perpendicular distance from node v to the origin-destination-line $\overleftrightarrow{OD}$, angle between $\overline{vO}$ and $\overline{OD}$, and node stretch relative to (O, D) pair. Note that symbolic regression finds a mathematical function (using a combination of an operator set $\{+, -, *, /, cos, sin, log, min, max, ...\}$) that best describes the relationship between the input features and the output on a given dataset. In addition to the distance from v to destination D that is used to achieve near-optimal routing in scale-free graphs, node stretch is found to be the dominant feature for predicting the Q-values using both decision tree and symbolic regression. These results motivate us to adopt these two features as the node states for the best forwarder prediction for APNSP.

Accordingly, we design the input features, including state and action features, as follows:

- State feature, $f_s(v)$. For a packet with its specified origin O and destination D at node v, the state features are the vectors with the elements below.

    1. **Distance to destination,** $d(v, D)$**:** the distance between v and D.

2. **Node Stretch,** $s(O, D, v) = \frac{d(O,v)+d(v,D)}{d(O,D)}$**:** the stretch of the indirect distance between O and D that is via v with respect to the direct distance between O and D.

- Action feature, $f_a(a) = f_s(u)$. The feature for the action that forwards a packet from v to $u \in nbr(v)$, $f_a(a)$ is chosen to be the same as the state feature of u, $f_s(u)$.

Henceforth, we consider learning with two different combinations of input features, one with only $d(v, D), d(u, D)$ and the other with both $d(v, D), d(u, D)$ and $s(O, D, v), s(O, D, u)$.

6.2 Provable Generalizability of Routing Policies

We begin with a sufficient condition for how a routing policy, π, learned from a seed graph $G^* = (V^*, E^*) \in \mathbb{G}$, where $\mathbb{G}$ is the set of all uniform random graphs, with select samples generated from a subset of nodes in V^*, can generalize over other (and potentially all) graphs $G \in \mathbb{G}$. To that end, we will show that if there exists a local ranking metric function m, considering node states from the neighborhood, which can approximate the per-hop routing decision made by the global metric function Q, taking into account the node states from the entire graph, then m can be learned with a limited set of training samples independent of the size of seed graph.

A basis for generalizability in this setting is the concept of a ranking metric for each node $v \in V$ with respect to each $u \in nbr(v)$, where $nbr(v)$ denotes the set of v's one-hop neighbors. Let $f_s : V \to \mathbb{R}^I$ be a map of $v \in V$ to its state features (of cardinality I) and let $f_a : V \to \mathbb{R}^J$ be a map of $u \in nbr(v)$ to its action features (of

cardinality J). We define a ranking metric $m\colon f_s \times f_a \to \mathbb{R}$ to be a *linear* or *non-linear* function over the input features associated with a node v and its neighbor u.

For notational convenience, given an ordering $\langle u_0, ..., u_d \rangle$ of all nodes in $nbr(v)$, let $X_v = \{\langle f_s(v), f_a(u_0)\rangle, ..., \langle f_s(v), f_a(u_d)\rangle\}$, denote the set of vectors for each corresponding neighbor $u_k \in nbr(v), 0 \leq k \leq d$. Also, let $Y_v = \{Q(v, u_0), ..., Q(v, u_d)\}$ denote the corresponding set of Q-values.

We next define a property, **MonotonicRankPres**, relating the local ranking metric m and the corresponding Q-values set Y_v, namely, the global ranking metric, that suffices to learn a DNN model for ranking the neighbors $u \in nbr(v)$ using local node states to achieve the same ranking using Q-values.

Definition 1. *For any node v in V, metric m_v, and Q-values set Y_v,* MonotonicRankPres *holds iff*

> *For any ordering $\langle u_0, ..., u_d \rangle$ of all nodes in $nbr(v)$,*
> *if $\langle (f_s(v), f_a(u_0)), \ldots, (f_s(v), f_a(u_d))\rangle$ is monotonically non-decreasing* [15] *,*
> *then both $\langle m_v(f_s(v), f_a(u_0)), \ldots, m_v(f_s(v), f_a(u_d)\rangle$ and $\langle Q(v, u_0), ..., Q(v, u_d)\rangle$*
> *are monotonically non-decreasing.*

Theorem 4 (Learnability)**.** *Let* MonotonicRankPres *hold for node v in V, metric m_v, and Q-values set Y_v. There exists a learnable DNN H, $H : \mathbb{R}^{I+J} \to \mathbb{R}$, with training samples $\langle X_v, Y_v\rangle$, that achieves optimal ranking of all $u \in nbr(v)$, i.e., its output for the corresponding neighbors of v, $\langle\, H(f_s(v), f_a(u_0)), ..., H(f_s(v), f_a(u_d))\,\rangle$, is monotonically non-decreasing.*

Proof. The theorem follows directly from a construction [68] that demonstrates the learnability of neural networks when there exists a monotonic mapping, m'_v, from the

[15] Let $\langle (f_s(v), f_a(u_0)), \ldots, (f_s(v), f_a(u_d)\rangle$ be $\langle (p_{0,0}, ..., p_{0,r}), \ldots, (p_{d,0}, ..., p_{d,r})\rangle$. By monotonic non-decreasing order in $\langle (f_s(v), f_a(u_0)), \ldots, (f_s(v), f_a(u_d)\rangle$, we mean $p_{0,k} \leq ... \leq p_{d,k}$ for all k, $0 \leq k \leq r$.

input vector to the output value: in this case, the input vector is $\langle f_s(v), f_a(u)\rangle$ and the output value is $Q(v,u)$. □

Next, we lift the sufficient condition to provide a general basis for ranking the neighbors of all nodes in a graph to predict the best forwarder, by training with only the samples derived from one (or a few) of its nodes. Subsequently, we further lift the sufficient condition for similarly ranking the neighbors of all graphs in $\mathbb{G}$.

Theorem 5 (Cross-Node Generalizability)**.** *For any graph G, $G=(V,E)$, if there exists a ranking metric $m(f_s(v), f_a(u))$ that satisfies the MonotonicRankPres property for all $v \in V$, then an optimal ranking policy for all $v \in V$ is learnable with only a subset of training samples $\langle X_{V'}, Y_{V'}\rangle$, where $V' \subseteq V$, $X_{V'} = \bigcup_{v\in V'} X_v$, and $Y_{V'} = \bigcup_{v\in V'} Y_v$.*

Proof. Our objective is to learn from a sample subset $\langle X_{V'}, Y_{V'}\rangle$ a function, $m' : f_s \times f_a \rightarrow \mathbb{R}$, that satisfies MonotonicRankPres property for all nodes in a graph.

Since there exists $m()$ for which MonotonicRankPres holds for all nodes, then from Theorem 4, a function $m' : f_s \times f_a \rightarrow \mathbb{R}$ can be learned from a subset of nodes V', $V' \subseteq V$ and $|V'| \geq 1$. In fact, it suffices to learn $m'()$ with the training samples derived from a single v with the highest degree (degree$(v) > 1$)[16].

It remains to show that m' satisfies MonotonicRankPres on nodes $v'' \in V \setminus V'$.Without loss of generality, let us assume that $Q(v'', u''_0)$, $Q(v'', u''_1)$, ... are in monotonic non-decreasing order. Since $m'()$ is monotonic non-decreasing by construction, for this neighbor ordering, by definition, the $m'()$ values will also be in the same order as that of Q. □

[16]Let δ_{max} be the maximum node degree in a graph. The order of the (polynomial) function m' to satisfy MonotonicRankPres is at most $\delta_{max} - 1$.

Note that if the $Q(v,u)$ value corresponds to the optimal (shortest) path Q-value for each (v,u) pair, then the DNN indicated by Theorem 5 achieves an optimal routing policy for all nodes in V. Note also that if the ranking metric m satisfies *MonotonicRankPres* not for all nodes but for almost all nodes, a policy learned from samples from one or more nodes v that satisfy *MonotonicRankPres* may not achieve optimal routing for all nodes. Nevertheless, if the relative measure of the number of nodes that do not satisfy *MonotonicRankPres* to the number of nodes that do satisfy *MonotonicRankPres* is small, then with high probability the policy achieves near-optimal routing.

Theorem 6 (Cross-Graph Generalizability)**.** *If there exists a ranking metric* $m(f_s(v), f_a(u))$ *that satisfies MonotonicRankPres property for the nodes in all graphs* $G \in \mathbb{G}$*, then an optimal ranking policy is learnable by using training samples from one or more nodes in one or more chosen seed graph(s)* $G^* \in \mathbb{G}$*.*

Proof. If there exists $m()$ that satisfies MonotonicRankPres property for the nodes in all graphs, then a function $m' : f_s \times f_a \rightarrow \mathbb{R}$ can be learned by using training samples from one or more nodes in one or more chosen seed graph(s) G^*. It is sufficient to learn $m'()$ with the training samples derived at least from one node with the highest degree in the seed graph G^*. $m'()$ achieves the optimal ranking policy. □

Again, if Theorem 6 is considered in the context of Q-values corresponding to optimal shortest paths, the learned routing policy π generalizes to achieving optimal routing over all graphs $G \in \mathbb{G}$. And if we relax the requirement that *MonotonicRankPres* holds for all nodes of all graphs in $\mathbb{G}$ to only require that for almost all graphs $G \in \mathbb{G}$, there is a high similarity between the ranking orders of neighbors using

m and the ones using optimal Q-value over all nodes v, then with high probability the policy achieves near-optimal routing.

We now instantiate the theory for provable generalizability for APNSP.

Proposition 1. *For APNSP, there exists a local ranking metric* $m_1(f_s(v), f_a(u))$ *of the form* $w_1.d(v, D) + w_2.d(u, D)$ *based on the* **distance** *input feature that satisfies MonotonicRankPres property for almost all nodes in almost all graphs* G.

Also, there exists a local ranking metric $m_2(f_s(v), f_a(u))$ *of the form* $w_1.d(v, D) + w_2.s(O, D, v) + w_3.d(u, D) + w_4.s(O, D, u)$ *based on both* **distance** *and* **node stretch** *input features that satisfies MonotonicRankPres property for almost all nodes in almost all graphs* G.

We empirically validate Proposition 1. MonotonicRankPres is quantified in terms of Ranking Similarity; high ranking similarity implies that MonotonicRankPres holds with high probability. We show that Proposition 1 holds for both Euclidean and hyperbolic spaces with respectively chosen weights w for m_1 and m_2.

It follows that an efficient generalizable policy for APNSP is feasible for each of the two chosen input feature sets, given the existence of respective ranking metrics that with high probability satisfy MonotonicRankPres. For APNSP, the optimal $Q(v, u)$ values can be retrieved by calculating the length of the shortest path starting from v toward u until reaching the destination. A near-optimal routing policy may then be learned via supervised learning on a single seed graph.

6.3 Single Graph Learning Algorithm

6.3.1 Ranking Similarity between Local and Global Metrics

Based on the theory of Section 6.2, we now investigate the existence of a ranking metric m for the first input feature, as well as the pair of input features, and demonstrate that in both cases the ranking similarity is close to 1 across nodes in almost all graphs G, regardless of their size and density. The results thus empirically corroborate the feasibility of routing policy generalization from single graph learning, and also guide the selection of seed graphs and training samples.

Let $SIM_v(m, Q^*) \in [0, 1]$ denote the *ranking similarity* between the ranking metric m and the optimal Q-values (Q^*) at node v. Moreover, let $SIM_G(m, Q^*) \in [0, 1]$ denote the average ranking similarity between m and Q^* across all nodes $v \in V$ in a given graph $G = (V, E)$, i.e., $SIM_G(m, Q^*) = \frac{\sum_{v \in V} SIM_v(m, Q^*)}{|V|}$.

Conceptually, $SIM_v(m, Q^*)$ should tend to 1 as the order of neighbors in the sorted ascending order of $m(f_s(v), f_a(u)), u \in nbr(v)$, comes closer to matching the order of the neighbors in the sorted ascending order of $Q^*(v, u)$. More specifically, we adopt Discounted Cumulative Gain (DCG) [37] to formally define the ranking similarity. The idea of DCG is to evaluate the similarity between two ranking sequences by calculating the sum of graded relevance of all elements in a sequence. First, a sorted sequence of $Q^*(v, u)$ with length $L = |nbr(v)|$, $u \in nbr(v)$, is constructed as the ideal ranking, denoted by A. For each position i in A, we assign a graded relevance $rel_A[i]$ by following the rule: $rel_A[i] = (L - i)^2, i = 0...L - 1$[17]. The value of DCG

[17]The assignment of graded relevance could be any way to decrease the value from left to right positions. Here we use squared value to assign dominant weights to the positions close to leftmost.

accumulated at a particular rank position τ is defined as:

$$DCG_\tau = \sum_{i=1}^{\tau} \frac{rel[i]}{\log_2(i+1)}.$$

For example, let $A = [4, 1, 3, 2, 5]$. The corresponding $rel_A[i]$ and $\frac{rel_A[i]}{\log_2(i+1)}$ values are shown in Table 6.1. Then A's $DCG_3 = 25 + 10.095 + 4.5 = 39.595$.

Table 6.1: An example of DCG calculation for an ideal ranking $A = [4, 1, 3, 2, 5]$.

i	$A[i]$	$rel_A[i]$	$\log_2(i+1)$	$\frac{rel_A[i]}{\log_2(i+1)}$
1	4	25	1	25
2	1	16	1.585	10.095
3	3	9	2	4.5
4	2	4	2.322	1.723
5	5	1	2.807	0.387

Table 6.2: An example of DCG calculation for an estimated ranking $B = [1, 2, 4, 5, 6]$.

j	$B[j]$	$rel_B[j]$	$\log_2(j+1)$	$\frac{rel_B[j]}{\log_2(j+1)}$
1	1	16	1	16
2	2	4	1.585	2.524
3	4	25	2	12.5
4	5	1	2.322	0.431
5	6	0	2.807	0

Next, a sorted sequence of $m(f_s(v), f_a(u))$ with length $L = |nbr(v)|$, $u \in nbr(v)$, is constructed as the estimated ranking, denoted by B. Let $B = [1, 2, 4, 5, 6]$. The

graded relevance for $B[j]$ depends on the position of $B[j]$ in A and follows the rule:

$$rel_B[j] = \begin{cases} rel_A[i], & if\ (\exists i\ :\ A[i] = B[j]) \\ 0, & otherwise \end{cases}$$

Then B's corresponding $rel_B[j]$ and $\frac{rel_B[j]}{\log_2(j+1)}$ values are shown in Table 6.2. Accordingly, B's $DCG_3 = 16 + 2.524 + 12.5 = 31.024$. The ranking similarity between B and A is calculated by the ratio of B's DCG to A's DCG, i.e., $\frac{31.024}{39.595} = 0.784$.

6.3.2 Selection of Seed Graph and Graph Subsamples

To achieve both cross-graph generalizability and cross-node generalizability, we develop a knowledge-guided mechanism with the following two selection components: (1) seed graph selection and (2) graph subsample selection.

Seed Graph Selection

The choice of seed graph depends primarily on the analysis of cross-node generalizability (Theorem 5) across a sufficient set of uniform random graphs with diverse sizes and densities/average node degrees. In Figures 6.8 and 6.9, we empirically show that, with the use of distance and node stretch, a good seed graph (with high $SIM_G(m, Q^*)$) is likely to exist in a set of graphs with modest size (e.g., 50) and high density (e.g., 5) in the Euclidean space and high average node degree (e.g., 4) in the hyperbolic space.

There may be applications where analysis of large (or full) graphs is not always possible. In such situations, given a graph G, an alternative choice of seed graph can be from a small subgraph $G' = (V', E'), V' \subset V, E' \subset E$ with relatively high cross-node generalizability. Note that, in Theorem 5, for a graph $G = (v, E)$ satisfying the *MonotonicRankPres* property for all $v \in V$, *MonotonicRankPres* still holds for

$v' \in V'$ in a subgraph $G' = (V', E')$ of G. This is because the learnable function m still preserves the optimal routing policy for all nodes in a subset of $nbr(v)$.

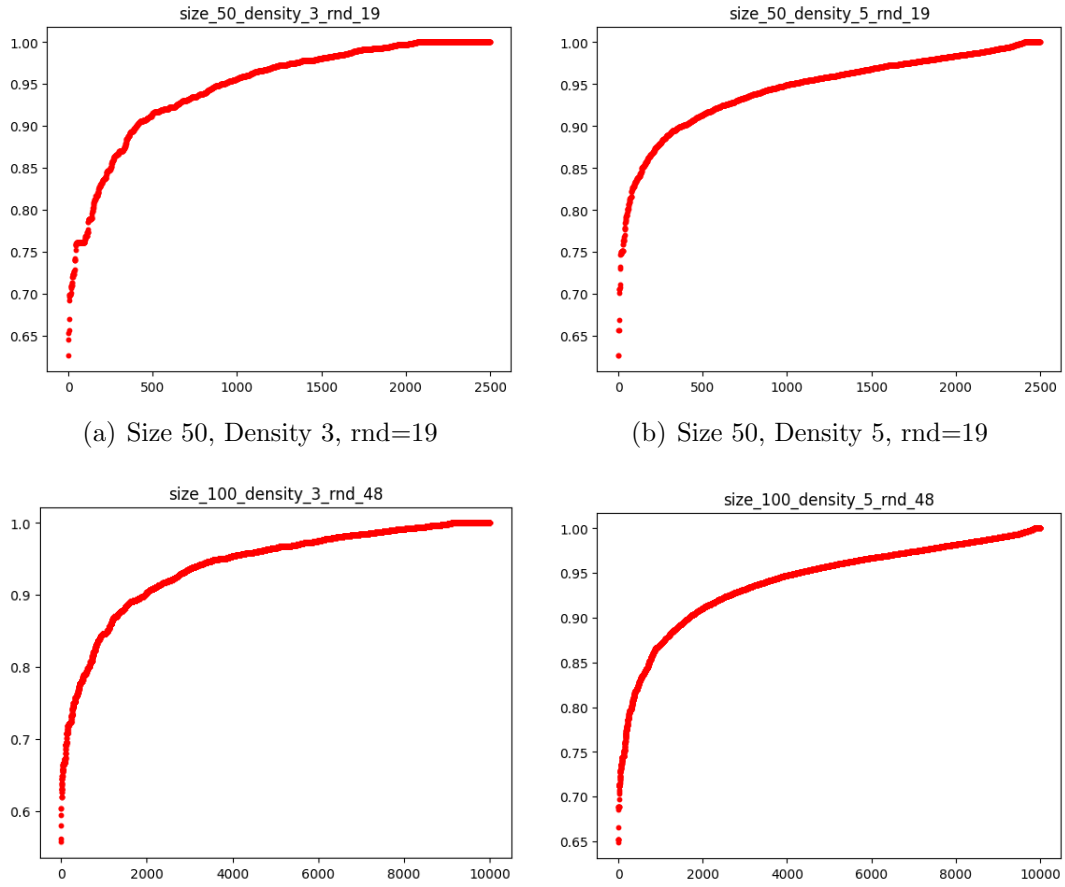
Graph Subsamples Selection. We provide the following scheme of subsample selection for a given graph $G = (V, E)$ to choose a set of ϕ nodes for generating $\phi\delta$ training samples.

1. Select an origin and destination pair $(O, D), O, D \in V$.
2. Select ϕ nodes, $v_0, ..., v_{\phi-1} \in V \setminus D$.
3. For each chosen node $v_0, ..., v_{\phi-1}$, respectively, collect the subsamples $\langle X, Y \rangle$, where
   $X = \bigcup_{v \in \{v_0,...,v_{\phi-1}\} \wedge u \in nbr(v)} \{\langle f_s(v), f_a(u) \rangle\}$ and
   $Y = \bigcup_{v \in \{v_0,...,v_{\phi-1}\} \wedge u \in nbr(v)} \{Q^*(v, u)\}$.

Ranking similarity with distance input feature. For $\langle f_s(v), f_a(u) \rangle = \langle d(v, D), d(u, D) \rangle$, we examine if there exists a function m that yields high $SIM_v(m, Q^*)$ and $SIM_G(m, Q^*)$ across different network configurations. Let the ranking metric m be $m(f_s(v), f_a(u)) = -d(u, D)$. (Recall that $Q^*(f_s(v), f_a(u))$ is the cumulative negative length of the shortest path.)

In Figure 6.2, we plot $SIM_v(m, Q^*)$ for all $v, D \in V$ for a given random Euclidean graph with size 50 and density in {3, 5}. In Figure 6.3, we plot $SIM_v(m, Q^*)$ for all $v, D \in V$ for a given random hyperbolic graph with size 50 and average node degree in {2, 4}. Each point represents the value of $SIM_v(m, Q^*)$ for a given v and D. All $|V|^2$ points are shown in ascending order. The sub-figures in Figures 6.2 and 6.3 illustrate that at least 75% of the points have high similarity ($\geq$ 90%) between m and Q^*. According to Theorem 5, the distribution of $SIM_v(m, Q^*)$ implies

that training samples collected from one (or a few) nodes in this large set can be sufficient to learn a near-optimal routing policy that achieves high accuracy across nodes. On the other hand, using training samples generated from the nodes with relatively low $SIM_v(m, Q^*)$ should be avoided. This observation motivates a knowledge-guide mechanism to carefully choose the data samples used for training.



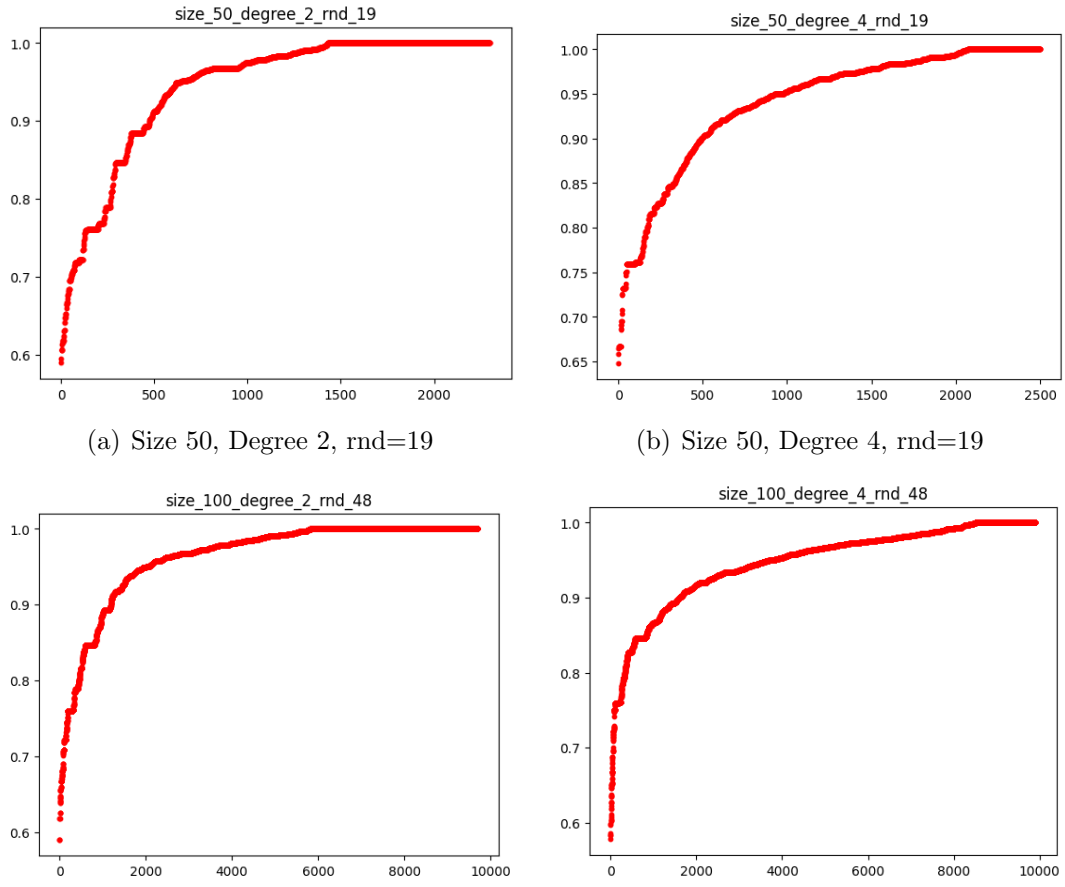
(a) Size 50, Density 3, rnd=19

(b) Size 50, Density 5, rnd=19

(c) Size 100, Density 3, rnd=48

(d) Size 100, Density 5, rnd=48

Figure 6.2: Distribution of $SIM_v(m, Q^*)$ given a uniform random graph in the Euclidean space with random seed (rnd), where m is the ranking metric for distance $d(u, D)$.



(a) Size 50, Degree 2, rnd=19

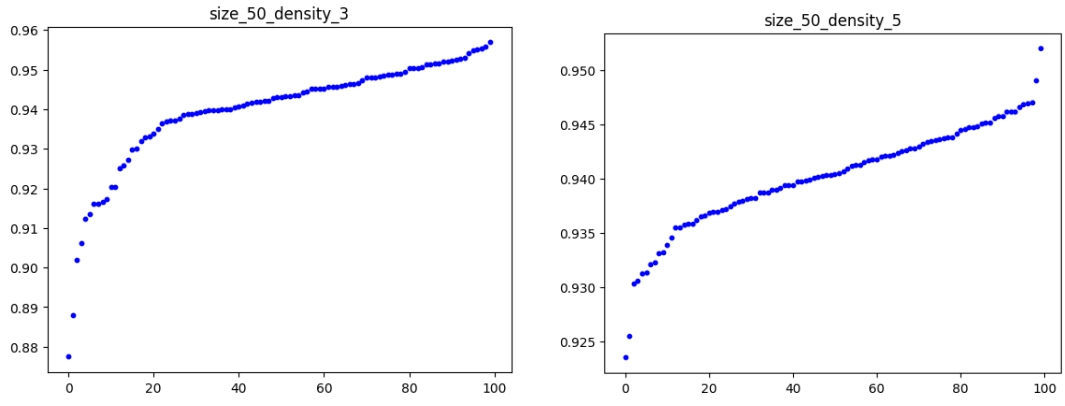
(b) Size 50, Degree 4, rnd=19

(c) Size 100, Degree 2, rnd=48

(d) Size 100, Degree 4, rnd=48

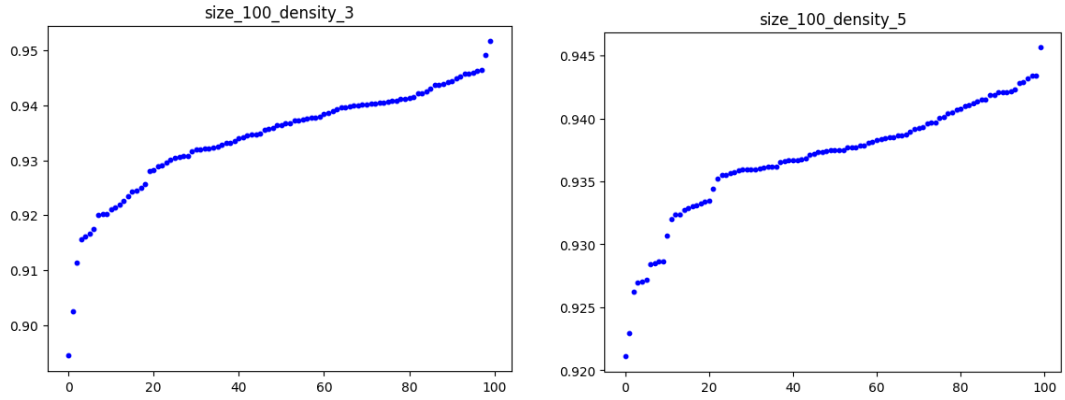
Figure 6.3: Distribution of $SIM_v(m, Q^*)$ given a uniform random graph in the hyperbolic space with random seed (rnd), where m is the ranking metric for distance $d(u, D)$.

In Figures 6.4 and 6.5, we show the distribution of $SIM_G(m, Q^*)$ in ascending order for a set of 100 graphs, respectively with size 50 and density in {3, 5} in the Euclidean space and average node degree in {2, 5} in the hyperbolic space. Note that in high-density Euclidean graphs (say density 5) and high-degree hyperbolic graphs (say degree 4), all 100 graphs have high similarity ($\geq$ 90%) between m and Q^*, implying that training with samples from almost any high-density Euclidean graph



(a) Size 50, Density 3

(b) Size 50, Density 5



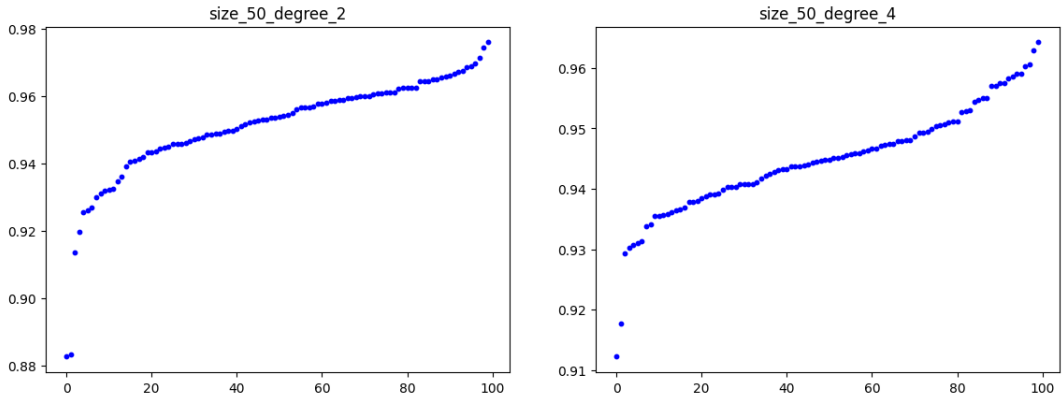
(c) Size 100, Density 3

(d) Size 100, Density 5

Figure 6.4: Distribution of $SIM_G(m, Q^*)$ across 100 uniform random graphs in the Euclidean space, where m is the ranking metric for distance $d(u, D)$.

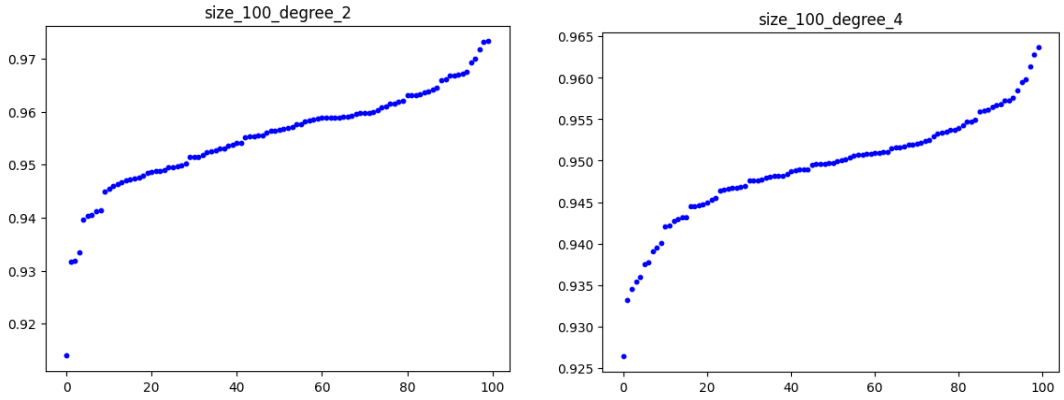
and high-degree hyperbolic graph can be sufficient to learn a routing policy with high performance. On the other hand, in low-density Euclidean graphs (say density 3) and low-degree hyperbolic graphs (say degree 2), there exist a small set of graphs with slightly low $SIM_G(m, Q^*)$, pointing to the importance of careful selection of seed graph(s) for training.

According to the results in Figures 6.2 and 6.3 and Figures 6.4 and 6.5, using distance implies the existence of a ranking metric that should with high probability



(a) Size 50, Degree 2

(b) Size 50, Degree 4



(c) Size 100, Degree 2

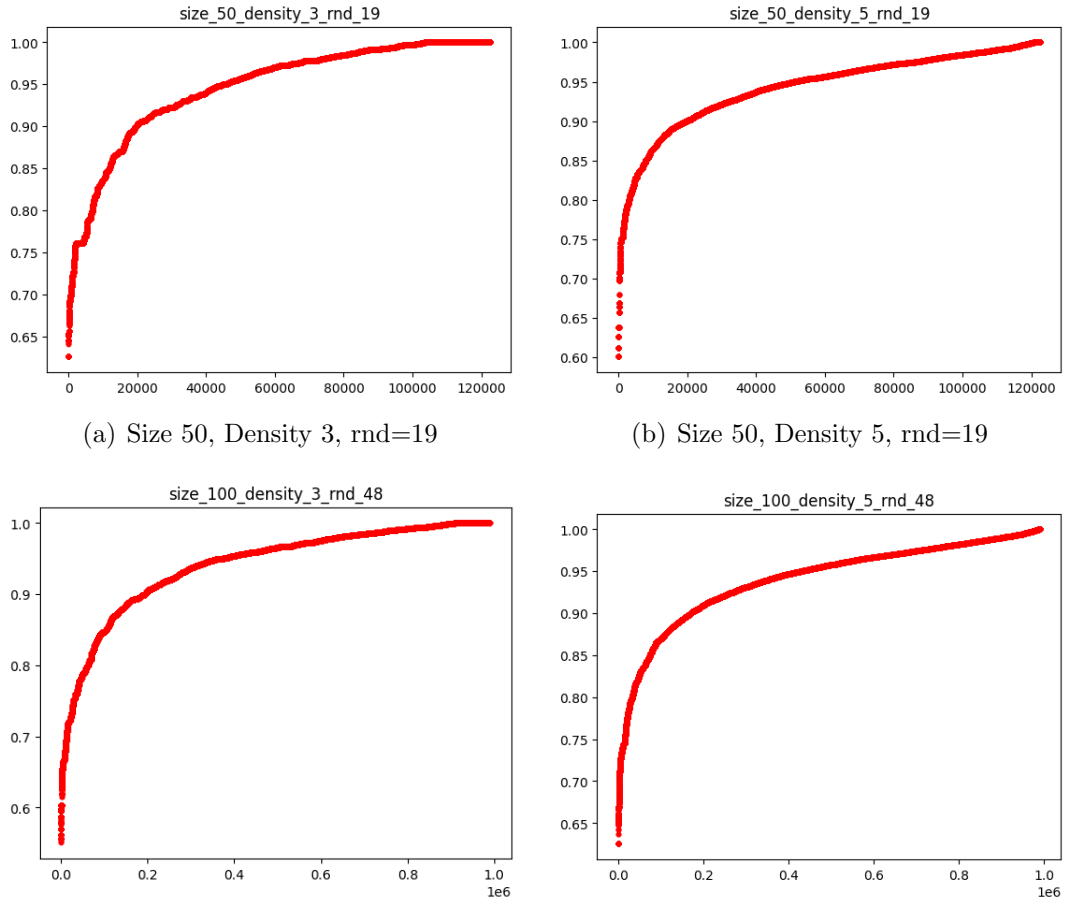
(d) Size 100, Degree 4

Figure 6.5: Distribution of $SIM_G(m, Q^*)$ across 100 uniform random graphs in the hyperbolic space, where m is the ranking metric for Euclidean distance $d(u, D)$.

satisfy cross-node generalizability and cross-graph generalizability. The ranking function, $m(f_s(v), f_a(u)) = -d(u, D)$, or an analogue can be easily learned by a DNN, and then the learned routing policy should achieve high performance across uniform random graphs.

Ranking similarity with distance and node stretch input features.

We examine if there exists a function m that yields high $SIM_v(m, Q^*)$ and $SIM_G(m, Q^*)$ across different network configurations, for $\langle f_s(v), f_a(u) \rangle =$



(a) Size 50, Density 3, rnd=19

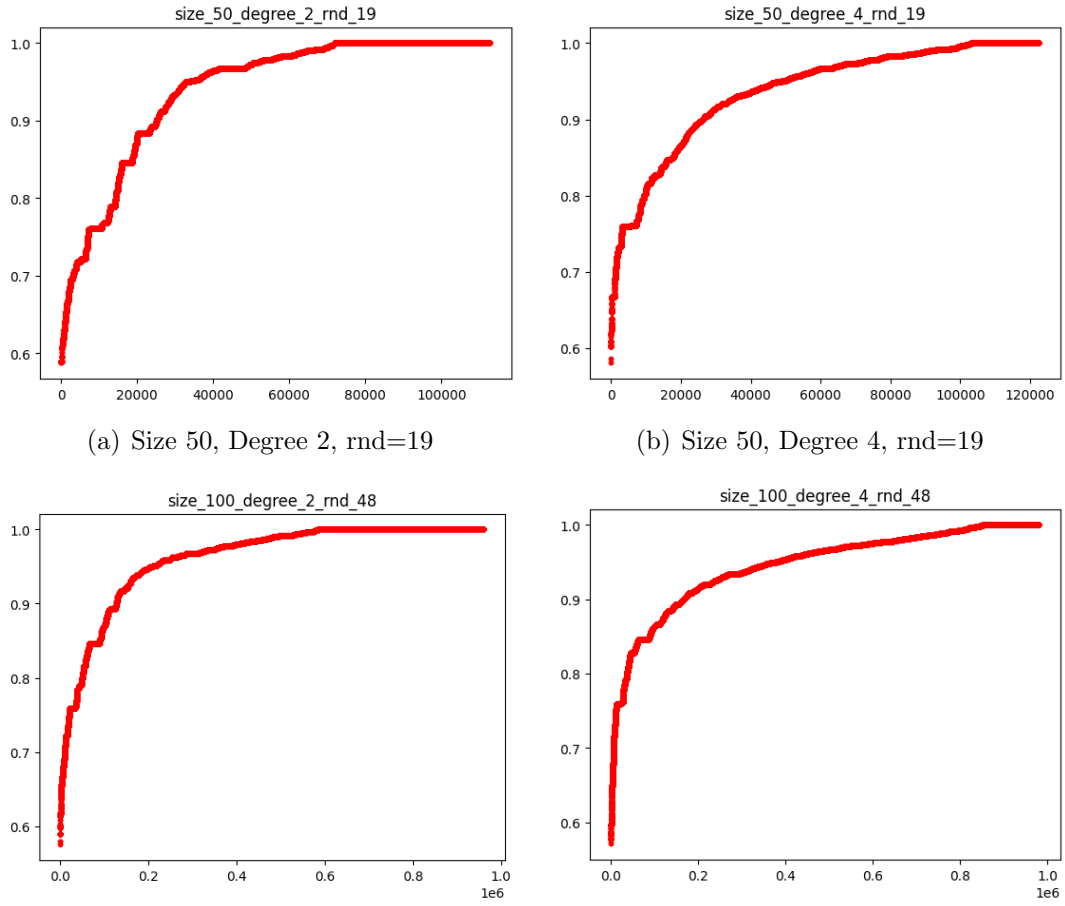
(b) Size 50, Density 5, rnd=19

(c) Size 100, Density 3, rnd=48

(d) Size 100, Density 5, rnd=48

Figure 6.6: Distribution of $SIM_v(m, Q^*)$ given a uniform random graph in the Euclidean space with a random seed (rnd), where m is the ranking metric for Euclidean distance $d(u, D)$ and node stretch $s(O, D, u)$.

$\langle d(v, D), s(O, D, v), d(u, D), s(O, D, u)\rangle$. Let the ranking metric m as $m(f_s(v), f_a(u)) = -0.875d(u, D) - 0.277s(O, D, u)$ in the Euclidean graphs and let m be $m(f_s(v), f_a(u)) = -d(u, D) - s(O, D, u)$ in the hyperbolic graphs. Note that $f_s(v)$ does not affect the sorting of $m(f_s(v), f_a(u))$. For convenience, we assign zero weight for $d(v, D)$ and $s(O, D, v)$ in m.



(a) Size 50, Degree 2, rnd=19

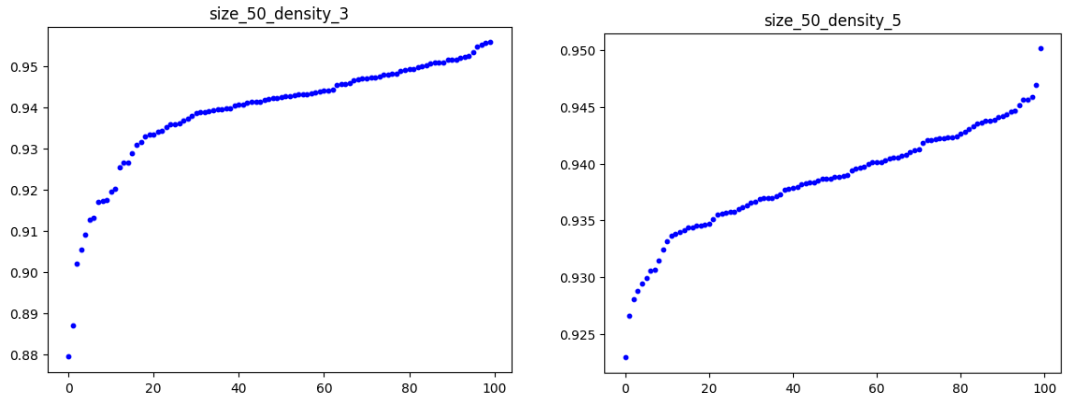
(b) Size 50, Degree 4, rnd=19

(c) Size 100, Degree 2, rnd=48

(d) Size 100, Degree 4, rnd=48

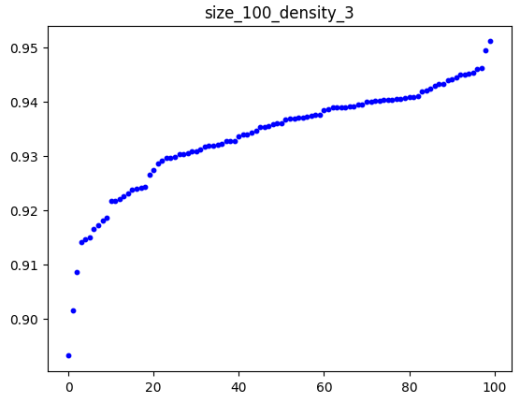
Figure 6.7: Distribution of $SIM_v(m, Q^*)$ given a uniform random graph in the hyperbolic space with a random seed (rnd), where m is the ranking metric for Euclidean distance $d(u, D)$ and node stretch $s(O, D, u)$.

In Figures 6.6 and 6.8, we apply the same network configurations in the Euclidean graphs as in Figures 6.2 and 6.4 to plot the distribution of $SIM_v(m, Q^*)$ and $SIM_G(m, Q^*)$ for the proposed choice of m. In Figures 6.7 and 6.9, we apply the same network configurations in the hyperbolic graphs as in Figures 6.3 and 6.5. Note that because the node stretch relies on $v, O, D \in V$, there are $|V|^3$ points plotted in Figureg 6.6 and 6.7 with ascending order. The sub-figures in Figures 6.6 and 6.7

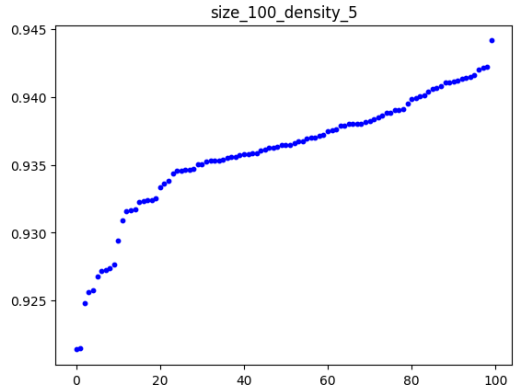


(a) Size 50, Density 3

(b) Size 50, Density 5



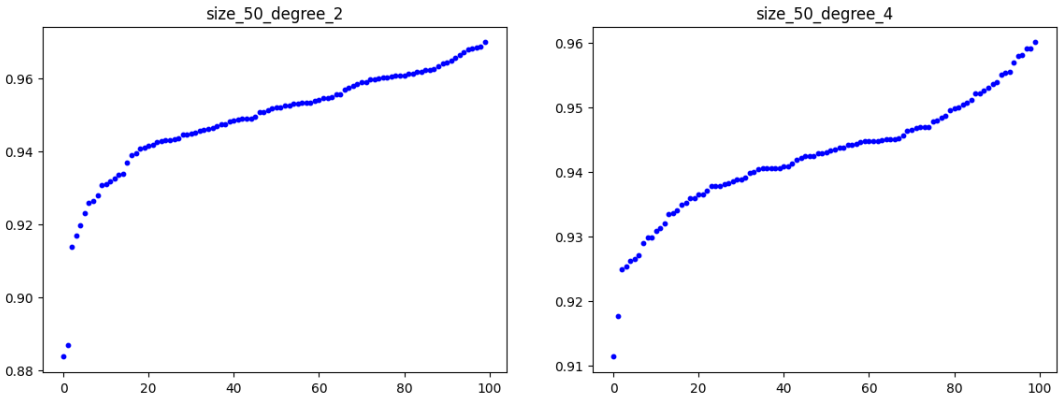
(c) Size 100, Density 3



(d) Size 100, Density 5

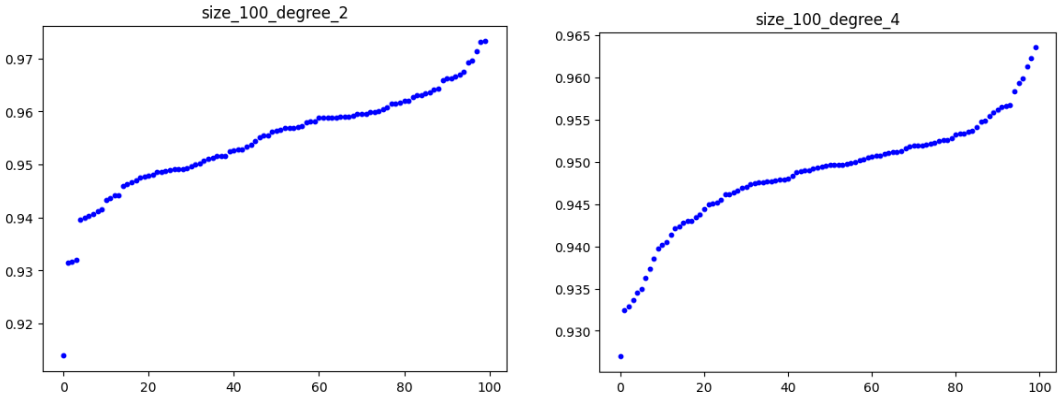
Figure 6.8: Distribution of $SIM_G(m, Q^*)$ across 100 uniform random graphs in the Euclidean space, where m is the ranking metric for distance $d(u, D)$ and node stretch $s(O, D, u)$.

demonstrate that at least 80% of the points have similarity above 90%, which is higher than the corresponding percentage of points in Figures 6.2 and 6.3. These observations indicate that using both distance and node stretch assures the existence of ranking metric, and in turn implies learnability of a DNN that with high probability achieves even better cross-node generalizability, compared to the one using only Euclidean distance in the input features.



(a) Size 50, Degree 2

(b) Size 50, Degree 4



(c) Size 100, Degree 2

(d) Size 100, Degree 4

Figure 6.9: Distribution of $SIM_G(m, Q^*)$ across 100 uniform random graphs in the hyperbolic space, where m is the ranking metric for distance $d(u, D)$ and node stretch $s(O, D, u)$.

Figures 6.8 and 6.9 shows a similar distribution of $SIM_G(m, Q^*)$ to Figures 6.4 and 6.5, respectively. In order to achieve cross-graph generalizability, we need to carefully select a seed graph for training, especially from graphs with moderate size and high density. Also, instead of using all the nodes to generate data samples, using a subset of nodes that avoid relatively low $SIM_v(m, Q^*)$ further improves the cross-node generalizability.

6.3.3 Supervised Learning for APNSP with Optimal Q-values

Given the dataset $\langle X, Y \rangle$ collected from a seed graph, we train a DNN based on supervised learning to capture the optimal ranking policy. Specifically, suppose the DNN H is parameterized by Θ. We seek to minimize the following loss function:

$$\min_{\Theta} \quad \sum_{\langle X,Y \rangle} \|H_{\Theta}(f_s(v), f_a(u)) - Q^*(v,u)\|^2. \tag{6.3}$$

Note that we assume that the optimal Q-values are known for the seed graph in the supervised learning above, which can be obtained based on the shortest path routing policies of the seed graph. By leveraging these optimal Q-values and supervised learning on the seed graph, a generalized routing policy is learned for APNSP routing over almost all uniform random graphs in both Euclidean and hyperbolic spaces, as we validate in the experiments in Section 6.4.

6.3.4 Reinforcement Learning for APNSP

For the case where the optimal Q-values of graphs are unknown, we solve the APNSP problem using RL. Using the same input features and seed graph selection procedure, the RL algorithm continuously improves the quality of Q-value estimations by interacting with the seed graph.

In contrast to the supervised learning algorithm, where we collect only a single copy of data samples from a set of chosen (shortest path) nodes once before training, in RL new training data samples from nodes in a shortest path, predicted by most recent training episode (i.e., based on the current Q-value estimation), are collected at the beginning of each training episode. Remarkably, the generalizability of the resulting RL routing policy across almost all uniform random graphs in Euclidean

and hyperbolic spaces is preserved. The details of the RL algorithm, named as RL-APNSP-ALGO, are shown in Algorithm 11.

Algorithm 11: RL-APNSP-ALGO

```
1  Input: nn: randomly initialized DNN; G*: seed graph; Φ: set of chosen
     nodes; O, D: chosen origin and destination
2  for episode = 1...EpiNum do
3  |  X:=[], Y:=[], i := 0;
4  |  for v ∈ Φ do
5  |  |  for u ∈ nbr(v) do
6  |  |  |  X[i] := ⟨f_s(v), f_a(u)⟩;
7  |  |  |  Estimate Q(v, u) using Equation 6.4;
8  |  |  |  Y[i] := Q(v, u);
9  |  |  |  i := i + 1;
10 |  |  end
11 |  end
12 |  for iter = 1...IterNum do
13 |  |  Train nn with ⟨X, Y⟩ based on Equation 6.5;
14 |  end
15 end
16 return nn;
```

More specifically, in Algorithm 11, Lines 2 to 15 outline the sample selection and training procedure for each episode. The for-loop in Lines 4 to 11 generates the training data samples from set of chosen nodes, where the data labels Y, i.e., target Q-values that we train the DNN to fit for improving the estimation accuracy of optimal Q-values, are given by:

$$Q^{target}(s,a) = r(s,a) + \gamma \max_{a'} Q(s',a'). \tag{6.4}$$

Next, based on the collected dataset, in Lines 12 to 14, the DNN is trained for a fixed number of iterations to minimize the following loss function:

$$\min_{\Theta} \quad \sum_{\langle X,Y \rangle} \|H_{\Theta}(f_s(v), f_a(u)) - Q^{target}(v,u)\|^2. \tag{6.5}$$

Since the target Q-values approach the optimal Q-values as the number of training episodes increases, minimizing Equation 6.5 will eventually lead to a learned model that nearly matches the supervised learning in Equation 6.3.

6.4 Evaluation

In this section, we discuss implementation of our machine learned routing policies and evaluate their performance in predicting all-pair near-shortest paths for graphs across different sizes and densities over Euclidean and hyperbolic spaces in Python3. We use PyTorch 2.0.1 on the CUDA 11.8 compute platform to implement DNNs as shown in Figure 6.1. Table 6.3 shows our simulation parameters for training and testing the routing policies.

6.4.1 Comparative Evaluation of Routing Policies

We compare the performance of the different versions of *Greedy Tensile* policies obtained using the following approaches:

- **Supervised (ϕ=3)**: Supervised learning from appropriately chosen seed graph G^* using graph subsamples selection with ϕ=3.
- **Supervised (all)**: Supervised learning from appropriately chosen seed graph G^* with samples generated from all nodes.

Table 6.3: Simulation Parameters

Symbol	**Meaning**	**Value**
N_{train}	size of seed graph	50
ρ_{train}	density of seed graph	5
N_{test}	sizes of tested graphs	27, 64, 125, 216
ρ_{test}	densities of tested graphs in Euclidean space	2, 3, 4, 5
δ_{test}	avg. node degree of tested graphs in hyperbolic space	1, 2, 3, 4
R	communication radius in Euclidean space	1000
$-\alpha$	negative curvature in hyperbolic space	-0.6
$\Omega = I + J$	# of input features	2,4
K	# of hidden layers	2
$N_e[]$	# of neurons in each hidden layer	$[50\Omega, \Omega]$
ϵ	margin for shortest paths prediction	0.05
ϕ	# of nodes for subsampling	3
γ	discount factor	1
$IterNum_S$	# of iterations in supervised learning	5000
$IterNum_{RL}$	# of iterations in RL	1000
$EpiNum$	# of episodes in RL	20

- **RL (ϕ=3)**: RL from appropriately chosen seed graph G^* using graph subsamples selection with ϕ=3.

- **RL (all)**: RL from appropriately chosen seed graph G^* with samples generated from all nodes.

- **GF**: Greedy forwarding that forwards packets to the one-hop neighbor with the minimum Euclidean distance to the destination.

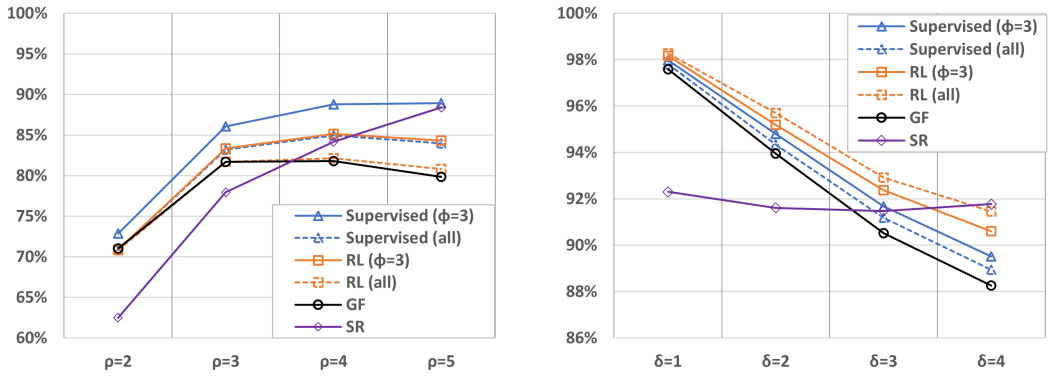
- **SR**: Symbolic Regression routing that learns a function, $s(O, D, u) + 0.64$, to forward packets to the one-hop neighbor with the minimum node stretch.

Note that for a given set of input features, both supervised learning and reinforcement learning schemes use the same DNN configuration to learn the routing policies. By using the subsampling mechanism, not only the sample complexity but also the training time will be significantly reduced in Supervised (ϕ=3) and RL (ϕ=3) compared to those in Supervised (all) and RL (all), respectively.

6.4.2 Zero-shot Generalization over Diverse Graphs

To evaluate the scalability and generalizability of the policies, we directly (i.e., without any adaptation) test the policies learned from the seed graph G^* on new uniform random graphs with different combinations of (N_{test}, ρ_{test}) in Euclidean spaces and $(N_{test}, \delta_{test})$ in hyperbolic spaces. We select 20 random graphs for each pair and calculate the average prediction accuracy over these $20{N_{test}}^2$ shortest paths.

For the DNNs with input $d(v, D)$ and $d(u, D)$, the tests confirm that the performance of all the learned policies exactly match the prediction accuracy of Greedy Forwarding in both Euclidean and hyperbolic spaces.



(a) Euclidean Space

(b) Hyperbolic Space

Figure 6.10: Average APNSP prediction accuracy across graph sizes with various ρ and δ for *Greedy Tensile* policies.

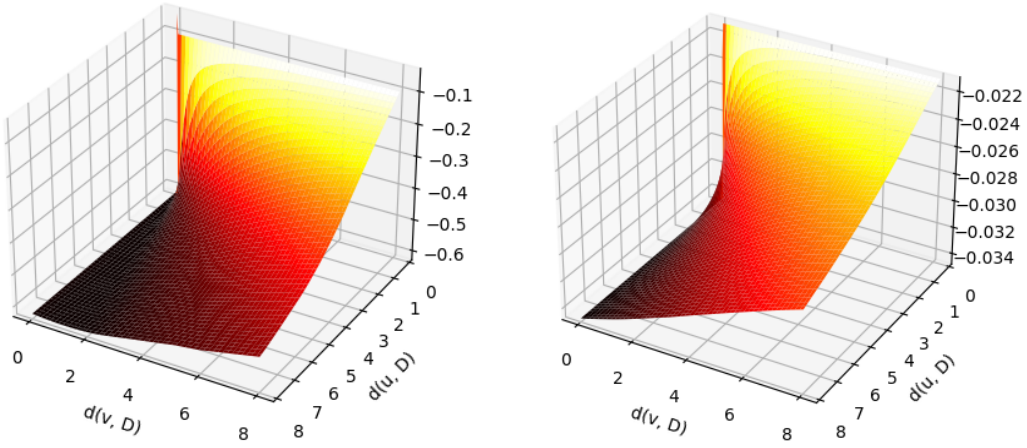
For the *Greedy Tensile* DNNs with input $\langle d(v,D), s(O,D,v), d(u,D), s(O,D,u)\rangle$, we plot in Figure 6.10 the respective average prediction accuracies across graph sizes in {27, 64, 125, 216} with density in {2, 3, 4, 5} in Euclidean space and average node degree in {1, 2, 3, 4} in hyperbolic space.

In Euclidean space, the Supervised (ϕ=3) approach achieves the best performance among all the approaches. In particular, compared to GF, the Supervised (ϕ=3) policy improves the accuracy up to 9% over GF, whereas the other learned policies show at least comparable performance in low-density graphs ($\rho = 2$) and achieve an improvement of up to 5% in graphs with $\rho \geq 3$. The performance gap between the DNNs and GF increases as the network density increases to a high level (e.g., $\rho = 5$), wherein GF was believed to work close to the optimal routing. SR, merely generalizing to a specific network configuration, yields an accuracy gap up to 9% over GF in low-density graphs ($\rho \leq 2$) and starts to outperform GF in high-density

graphs ($\rho \geq 4$). The performance of SR increases to come close to the performance of Supervised (ϕ=3) policy as the density increases to $\rho = 5$.

In hyperbolic space, the RL (all) policy improves the accuracy up to 3% over GF, whereas the other learned policies show at least comparable performance in low-degree graphs ($\delta = 1$) and achieve an improvement of up to 2% in graphs with $\delta \geq 2$. To the best of our knowledge, we are the first to provide routing policies that outperform GF in almost all random graphs; recall GF was shown to find almost optimal shortest path in scale-free topologies [62]. Again, SR only achieves comparable performance in high-degree graphs ($\delta \geq 3$) but incurs a significant performance gap to other policies in low-degree graphs ($\delta \leq 2$).

6.5 Symbolic Interpretability of Learned Model with Distance Input Features

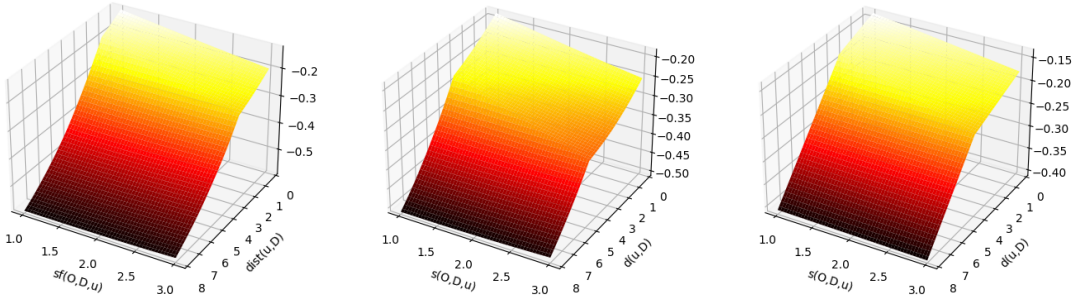


(a) $DNN, Euclidean$ (b) $DNN, hyperbolic$

Figure 6.11: The shape of ranking metrics of the learned DNN using $d(v, D), d(u, D)$ as the input features. The x and y axes represent $d(v, D)$ and $d(u, D)$, and the z axis is the ranking metric for routing.

Figure 6.11 shows that the learned DNN, using $d(v, D), d(u, D)$ as the input feature, is monotonically increasing as the $d(u, D)$ decreases for a fixed $d(v, D)$ in both Euclidean and hyperbolic spaces. Since $d(v, D)$ stay unchanged for a fixed routing node v at a given time and does not affect the ranking of $\langle m(f_s(O, D, v), f_a(O, D, u_0)), \ldots, m(f_s(O, D, v), f_a(O, D, u_d)) \rangle, u_i \in nbr(v)$, these models are effectively equivalently substituted for routing by a simple linear function (e.g., $-d(u, D)$) that achieves the same performance as Greedy Forwarding.

6.6 Symbolic Interpretability of Learned Model with both Distance and Node Stretch Input Features



(a) *Greedy Tensile DNN*, $d(v, D) = 2$ (b) *Greedy Tensile DNN*, $d(v, D) = 4$ (c) *Symbolic Approximation*

Figure 6.12: The shape of ranking metrics of the *Greedy Tensile* DNN and its two-linear action Symbolic Approximation policy in Euclidean space, given $s(O, D, v) = 1.2$. The x and y axes represent $s(O, D, u)$ and $d(u, D)$, and the z axis is the ranking metric for routing.

In this section, we symbolically interpret the learned model for *Greedy Tensile* routing, achieving a two orders of magnitude reduction in its operational complexity.

Figure 6.12 plots the output of its DNN, towards explaining the learned policy[18]. Since $d(v, D)$ and $s(O, D, v)$ stay unchanged for a fixed routing node v at a given time, we plot the shape of the ranking metric (z-axis) of the learned DNN according to varying $s(O, D, v)$ (x-axis) and $d(u, D)$ (y-axis) in the figure.

Figures 6.12(a) and 6.12(b) show that the *Greedy Tensile* DNN has two planes separated by a transition boundary that varies as $d(v, D)$ changes. The two planes can be respectively approximated by two different linear functions. The first function (for the upper plane) prefers both smaller $s(O, D, u)$ and $d(u, D)$. The second (for the lower plane) significantly prioritizes smaller $d(u, D)$. We find that the two functions that approximate the *Greedy Tensile* DNN can be symbolically represented by a guarded command:

$$z = \begin{cases} -0.01d(v, D) - 0.02s(O, D, u) - 0.01d(u, D) - 0.06, \\ \quad if\ d(u, D) < 1.02d(v, D) + 0.57s(O, D, u) - 0.69 \\ 0.03d(v, D) - 0.04d(u, D) - 0.15, \quad otherwise. \end{cases} \tag{6.6}$$

The weights of the guarded command are calculated using linear regression. Its first action assigns dominant weights to both $s(O, D, u)$ and $d(u, D)$, while its second action gives almost negligible weight to $s(O, D, u)$ compared to the weight of $d(uD)$. The shape of ranking metrics of the guarded command given $d(v, D) = 4$ is shown in Figure 6.12(c), which has a two-plane surface similar to Figure 6.12(b). Figure 6.13 shows that the accuracy of the two-linear-action policy using Equation 6.6 is close to that of the learned DNN.

The simplified two-linear-action policy also has substantially reduced operation complexity. Whereas the *Greedy Tensile* DNN requires at least $\Omega \times N_e[1] \times N_e[2](=$

[18]Since *Greedy Tensile* models in Euclidean and hyperbolic spaces have a similar shape, we only visualize the learned model in the Euclidean space here.

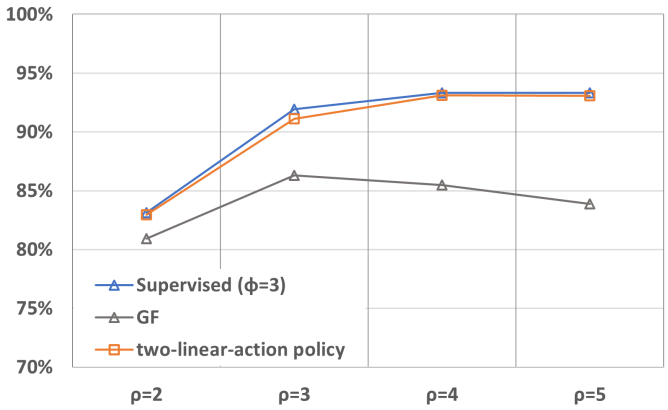


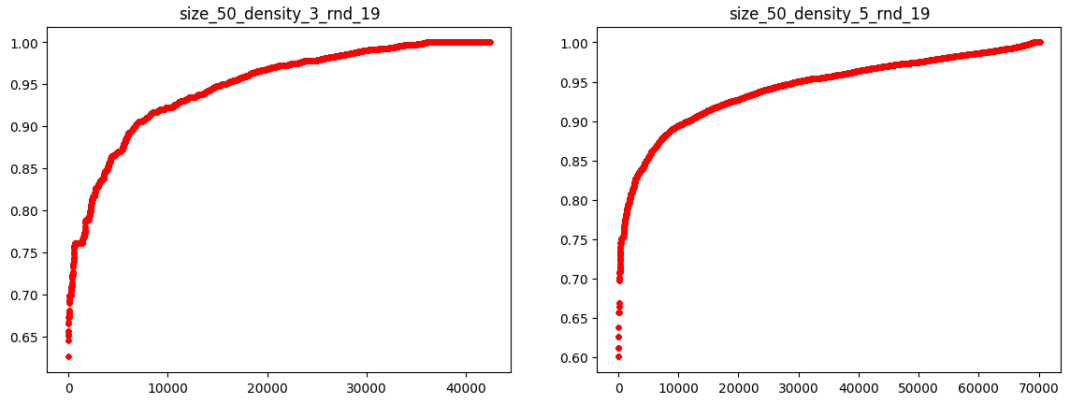
Figure 6.13: APNSP prediction accuracy for the two-linear-action policy in Equation 6.6 versus the policies of *Greedy Tensile* (Supervised ($\phi = 3$)) and greedy forwarding (GF) over graphs in the Euclidean space with size 50 and density in $\{2, 3, 4, 5\}$.

$4 * 200 * 4$) multiplications to output the Q-value for a given (state, action) pair, where Ω represents the number of input features of the DNN and $N_e[i]$ denotes the number of neurons in the i-th hidden layer, the two-linear-action policy needs less than ten multiplications.

6.7 Complexity of Graph Subsampling

According to the results in Section 6.3.2, seed nodes must have respectively have a high $SIM_v(m, Q^*)$, denoting the ranking similarity between the ranking metric m and the optimal Q-values (Q^*) at node v. The seed graph is assumed to have a sufficient number of nodes v that have high $SIM_v(m, Q^*)$.

In graph subsamples selection in Section 6.3.2, we search the ϕ nodes with highest values of $SIM_v(m, Q^*)$ for graph subsampling. Typically, depth-first search (DFS) or breadth-first search (BFS) can be used for searching a graph $G = (V, E)$ to get a node list of top-ϕ $SIM_v(m, Q^*)$ for subsampling. The time complexity of DFS and



(a) Size 50, Density 3, rnd=19, Λ=2 (b) Size 50, Density 5, rnd=19, Λ=2

Figure 6.14: Distribution of $SIM_v(m, Q^*)$ in the set of subgraph $G^{'}(D, \Lambda)$ of $G = (V, E)$ in Figure 6.6. Each subgraph $G^{'}(D, \Lambda)$ includes nodes only within Λ-hop distance to the given destination $D \in V$.

BFS is $O(|V| + |E|)$ since the worst case of traversal time in these two algorithms should explore every node and edge. However, searching an entire graph becomes time consuming as the number of nodes increase to a considerable value. Note that in a uniform random graph with an average node degree δ, the number of edges in a graph can be represented by $O(E) = O(\delta V)$.

To effectively reduce the search effort, an alternative is to search the nodes with top-ϕ $SIM_v(m, Q^*)$ values in a subgraph $G^{'}(D, \Lambda) = (V^{'}, E^{'}) \subseteq G$, where only nodes within Λ-hop distance to a given destination D are included in the subgraph.

Theorem 7 (Complexity of Graph Subsampling)**.** *Search ϕ nodes with top-ϕ $SIM_v(m, Q^*)$ in a subgraph $G^{'}(D, \Lambda) = (V^{'}, E^{'}) \subseteq G$ has a time complexity of $O(\delta^{\Lambda})$.*

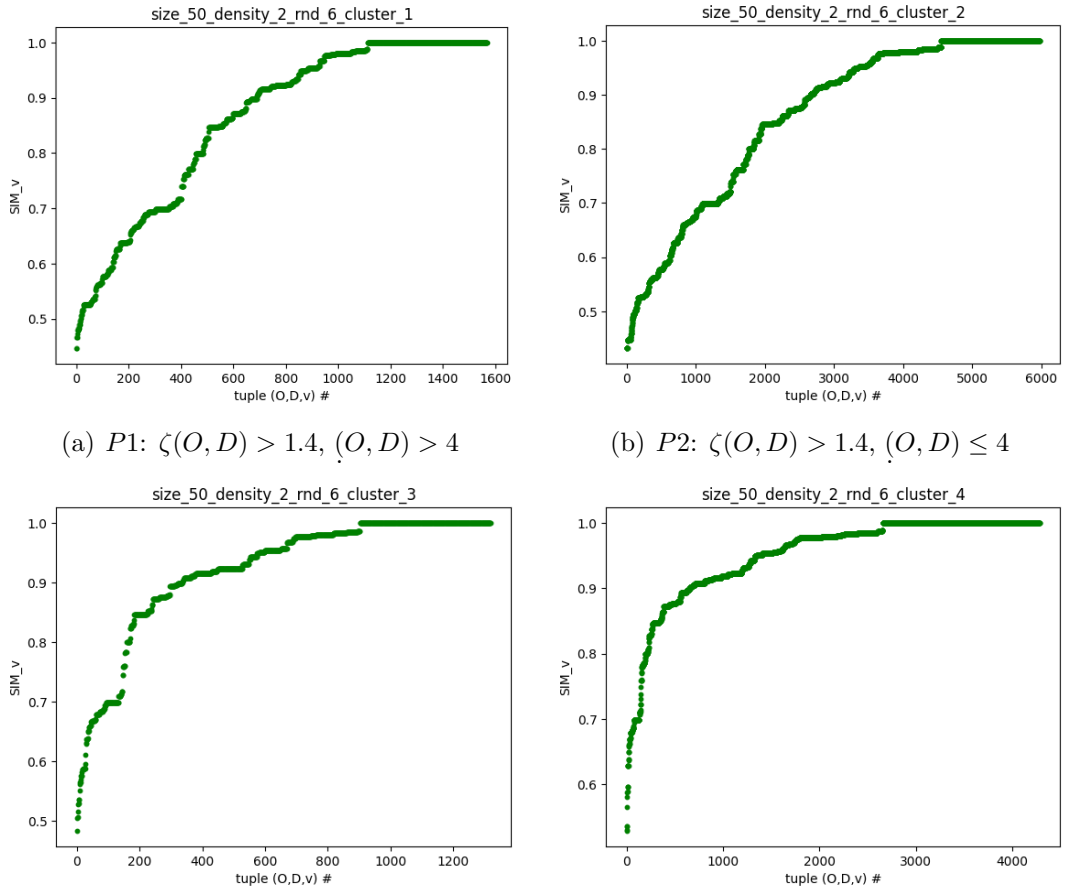
Proof. Figure 6.6 shows the distribution of $SIM_v(m, Q^*)$ for all pairs (v, O, D) in G and Figure 6.14 plot the distribution $SIM_v(m, Q^*)$ for the pairs $(v^{'}, O^{'}, D), v^{'}, O^{'} \in V^{'}$

in the set of subgraphs, $G^{'}(D,\Lambda) = (V^{'}, E^{'}), D \in V$. Figures 6.6 and 6.14 show a similar distribution of ranking similarity. Searching nodes with top-ϕ $SIM_v(m, Q^*)$ from a subgraph $G^{'}(D,\Lambda)$ does not impact the efficacy of subsampling for learning to achieve high accuracy. Therefore, for a given destination D in a random graph G with an average node degree δ, the time complexity of searching using DFS/BFS in the corresponding subgraph $G^{'}(D,\Lambda)$ is $O(\delta^{\Lambda})$, a constant number independent of the graph size. □

6.8 Clustering for Sparse Graphs

Even though our single-copy routing policy learned from a seed subgraph significantly improves prediction accuracy across random graphs compared to greedy forwarding, an appropriate routing policy detouring around holes may act differently from the one forwarding directly. We note that there may exist a small set of graphs, e.g., networks with a density lower than 2, where the learned policy receives only slightly better accuracy than greedy forwarding.

To further understand the routing performance in subgraphs with different path stretches in a sparse graph, we simply adopt a path stretch of 1.4 and a distance of 4 to separate the set of all origin-destination pairs in a graph into four partitions ($P1$, $P2$, $P3$, and $P4$). Fig. 6.15 shows the distribution of $SIM_v(m, Q^*)$ on each partition for comparing the local ranking using *Greedy Tensile* considering distance $d(u, D)$ and node stretch $s(O, D, u)$, $u \in nbr(v)$, and the global ranking using optimal Q-values. We note that, in $P1$ and $P2$, only about 60% of the points have similarity above 90%, whereas at least 80% of the points have similarity above 90% in $P3$ and



(a) $P1$: $\zeta(O,D) > 1.4$, $\dot{(}O,D) > 4$

(b) $P2$: $\zeta(O,D) > 1.4$, $\dot{(}O,D) \leq 4$

(c) $P3$: $\zeta(O,D) \leq 1.4$, $\dot{(}O,D) > 4$

(d) $P4$: $\zeta(O,D) \leq 1.4$, $\dot{(}O,D) \leq 4$

Figure 6.15: Distribution of $SIM_v(m, Q^*)$ on each **partition** in a graph with size 50 and density 2, where the local ranking metric m is calculated using *Greedy Tensile*.

$P4$. The result implies *Greedy Tensile* possibly performs not well for predicting the routing forwarder in a set of tuples (O, D, v) with high path stretch ($\zeta(O,D) > 1.4$).

Fig. 6.16 further shows the accuracy on $P1 - P4$ for policies respectively learned from $P1 - P4$ and the single-copy routing policy (*Greedy Tensile*) as well as GF. *Greedy Tensile* achieves the best performance in partitions $P3$ and $P4$ with the tuples of low path stretch but receives worse performance compared to GF and the policy

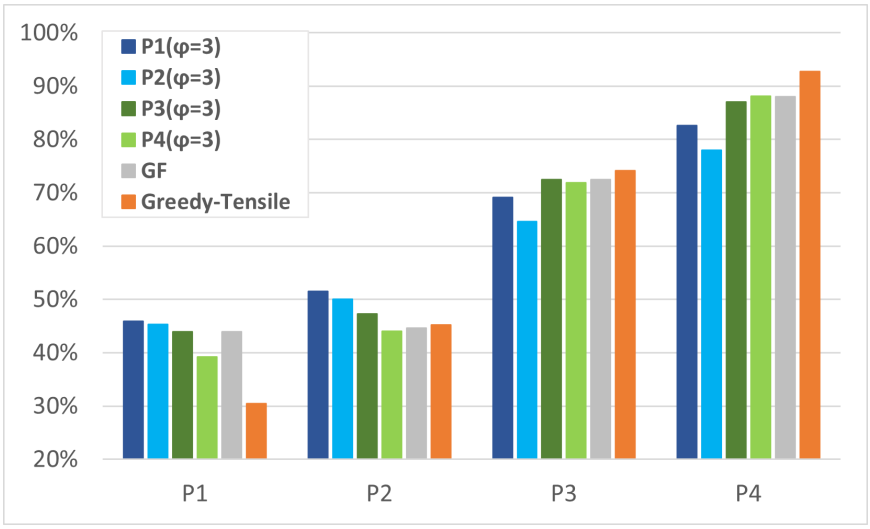


Figure 6.16: APNSP prediction accuracy for policies learned from four disjoint partitions in Figure 6.15, verses the routing policy learned from a seed graph G^* and GF.

learned from $P1$ and $P2$ in partitions $P1$ and $P2$ with the tuples of high path stretch. Intuitively, learning different single-copy policies for the partitions with different path stretches in sparse graphs can substantially improve the performance. For example, applying the policies $P1(\phi = 3)$ for $P1$ and $P2$, and using *Greedy Tensile* for $P3$ and $P4$ achieve the best performance in this graph.

To understand the preference of selecting routing forwarders in the policy $P1(\phi = 3)$, we symbolically interpret the learned model in Figure 6.17. Compared to the shape of *Greedy Tensile* preferring forwarders with both smaller $d(u, D)$ and $s(O, D, u)$ in Figure 6.12(b), the policy $P1(\phi = 3)$ prefers forwarders with smaller $d(u, D)$ but larger $s(O, D, u)$. Such a characteristic increases the chance to route packets to far forwarders from the straight line $\overline{OD}$ and, therefore, tends to yield a larger search region than that one of *Greedy Tensile*. According to the behavior, the policy $C1(\phi = 3)$ is named as *Greedy Lax*.

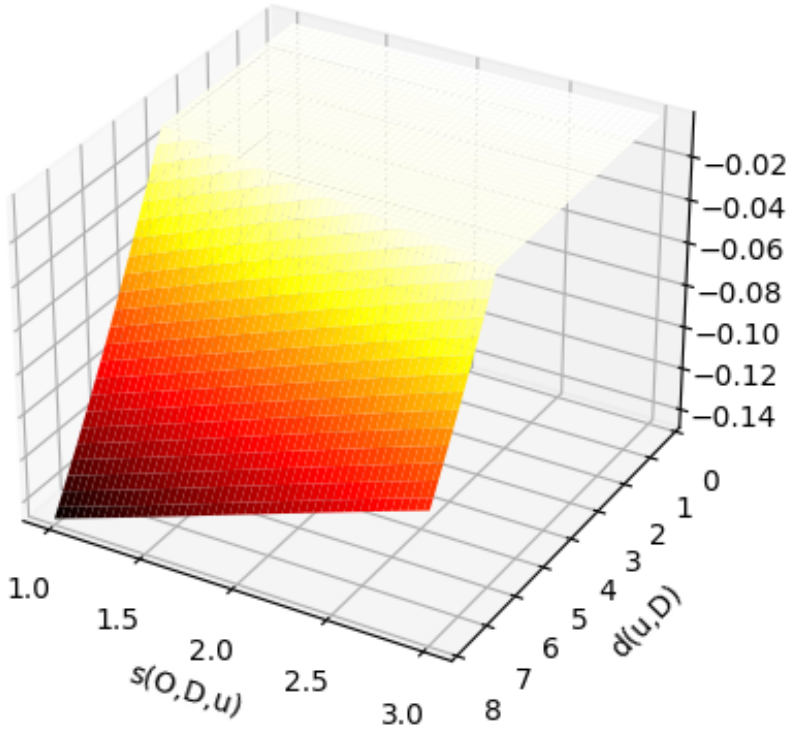


Figure 6.17: The shape of ranking metrics of the $C1(\phi = 3)$ (*Greedy Lax*) DNN in Euclidean space, given $s(O, D, v) = 1.2$ and $d(v, D) = 4$. The x and y axes represent $s(O, D, u)$ and $d(u, D)$, and the z axis is the ranking metric for routing.

In addition to applying a different single-copy policy for a partition, we are also motivated by Fig 6.15 to use multiple policies in a single partition. Recall that, in each partition, there is a negligible set of tuples where using *Greedy Tensile* as the local ranking metric has a relatively low correlation compared to the global ranking metric, $SIM_v(m, Q^*) \leq 0.9$. Therefore, there exist two sets of tuples in each partition or subgraph, where using *Greedy Tensile* to predict routing forwarders receives either high or low APNSP prediction accuracy. Accordingly, we separate the tuples into two sets: easy instances and hard instances. For easy instances, using *Greedy Tensile*

yields high accuracy for choosing the best forwarders and thus receives high APNSP prediction accuracy. For the hard instances, at least one routing policy other than *Greedy Tensile* needs to be learned to achieve better performance.

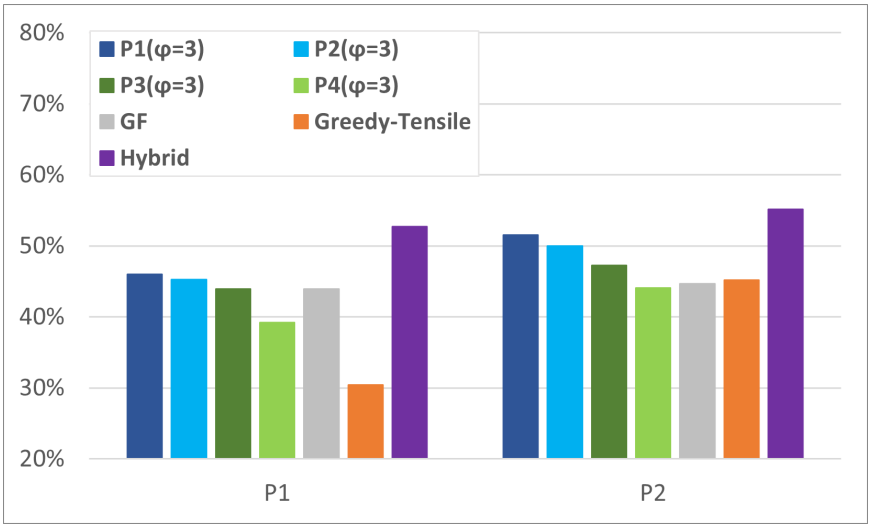
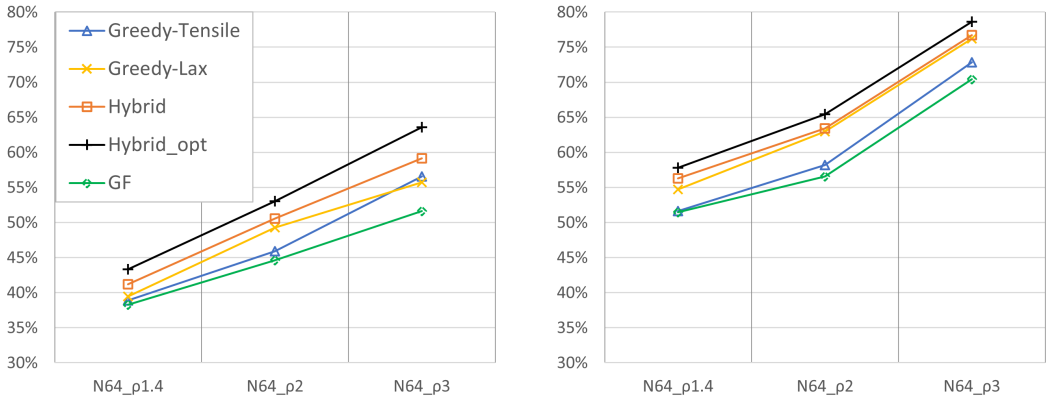


Figure 6.18: Comparison of APNSP prediction accuracy for hybrid and learned policies on partition $P1$ and $P2$ in Figure 6.16.

We propose a clustering method to differentiate the easy and hard instances by using $SIM_v(m, Q^*) = 0.9$ with m calculated using *Greedy Tensile* as the threshold, and then applying a different policy for each cluster. For the hybrid policy to predict the next forwarder for a given tuple (O, D, v), *Greedy Tensile* is chosen for high $SIM_v(m, Q^*) > 0.9$ to obtain high prediction accuracy, and *Greedy Lax* is used if the corresponding $SIM_v(m, Q^*)$ is equal or smaller than 0.9 to perform a different routing behavior. Figure 6.18 compares the APNSP prediction performance for the hybrid policy and the performance of all other policies shown in Figure 6.16 in $P1$ and $P2$, where *Greedy Tensile* works not well. Using hybrid policy further improves

the performance up to 7% from the best policy ($P1(\phi = 3)$) and up to 20% from *Greedy Tensile* in $P1$ and $P2$ in Figure 6.16.



(a) $P1$: $\zeta(O, D) > 1.4$, $\dot{(}O, D) > 4$ (b) $P2$: $\zeta(O, D) > 1.4$, $\dot{(}O, D) \leq 4$

Figure 6.19: Average APNSP prediction accuracy across random low $SIM_G(m, Q^*)$ graphs at size 64 in Euclidean space for hybrid and *Greedy Tensile* policies.

Fig. 6.19 shows that using hybrid policies outperforms the single-copy policies (*Greedy Tensile* and *Greedy Lax*) and receives up to 7% improvement from *Greedy Tensile* in the partitions $P1$ and $P2$ where *Greedy Lax* tends to outperform *Greedy Tensile* . The APNSP prediction accuracy is calculated across random sparse graphs $\rho \in 1.4, 2, 3$ with low $SIM_G(m, Q^*)$. The results show that *Greedy Lax* tends to receive better average APNSP prediction accuracy than *Greedy Tensile* in partitions $P1$ and $P2$ derived from sparse graphs, and the hybrid policy outperforms both these two single-copy policies across all configurations. Note that *Hybrid_opt* is used to demonstrate the upper bound of accuracy improvement for using hybrid policies between *Greedy Tensile* and *Greedy Lax* . *Hybrid_opt* is the hybrid policy that always chooses the correct policy between *Greedy Tensile* and *Greedy Lax* to obtain better optimal

Q-values. The result shows that the hybrid policy also preserves the generalizability across random graphs and dominates *Greedy Tensile* in those partitions.

6.9 Conclusions

We have shown that guiding machine learning with domain knowledge can lead to the rediscovery of a well-known routing policy (somewhat surprisingly), in addition to discovering a new routing policy, that performs well in terms of complexity, scalability, and generalizability. The theory we have presented in the paper is readily extended to other classes of graphs (such as non-uniform cluster distributions) and MDP actions that span multiple neighbors. Thus, albeit our illustration intentionally uses relatively familiar input features and local routing architectures, richer domain theory will be useful to guide machine learning of novel routing algorithms. Moreover, the routing policies are likely to be competitive for richer classes of graphs than the class of uniform random graphs on which we have focused our validation. For a small set of graphs (e.g., network density $\rho \leq 2$) where *Greedy Tensile* works not very well, we explore clustering to categorize the set of routing nodes into easy and hard instances. A hybrid policy taking advantage of two different routing behaviors is proposed to further improve the prediction accuracy for such graph typologies.

Chapter 7: Conclusions and Future Work

7.1 Conclusions

Our thesis provides a comprehensive solution to address the wireless mesh network optimization problems from deploying middle-mile networks to channel scheduling and packet routing in last-mile networks regarding throughput efficiency. In middle-mile network optimization, we propose the first MNO solution that works in polynomial time with an approximation ratio of $O(\ln |A| + \frac{|B|}{|A|-1} + \frac{|A|+|B|}{\gamma})$ in the tower and antenna cost. In last-mile networks or any networks following the model, we propose distributed protocols, QF-MAC and QF-Geo, that adapt across diverse network scenarios (e.g., of density, concurrency, mobility, etc.) and are robust to internal and external interference in multi-channel and multi-radio networks, where many extant protocol only performs well in a limited set of configurations. A well-planned network working with the two protocols appropriately utilizes the achievable capacity. We also propose a knowledge-guided learning algorithm for routing based on local search. It shows that machine learning techniques address the complexity, generalizability, and scalability issues in such networking problems. These contributions ease the difficulty of leveraging community-shared infrastructure for cost reduction and enhance

wireless communication performance to facilitate broadband connectivity in wireless mesh networks.

7.2 Discussion and Future Work

7.2.1 Flexible Design and Evaluation of Middle-mile Optimization Solution

Our solution can be further evaluated in targeted rural areas, such as southern Ohio, in partnership with local wireless network providers. The evaluation will include a cost analysis for likely scenarios of network growth that build on the capacity flexibility offered by our solution. In addition, a flexible framework can be proposed to change the topology adaptively (e.g., adding towers or adding/removing links) to support the increase of broadband demands as a long-term sustainability solution. A cost minimization problem will be introduced to change the network topology to satisfy the newly added demands and constraints.

Machine learning techniques enable the prediction of topologies that satisfy the constraints of planning networks. A trained model can be used to predict topologies in a complexity-efficient manner to adapt to changing demands and constraints. Specifically, the Steiner tree problem can be solved using graph neural networks and reinforcement learning approaches. As the throughput demands vary over spatial and temporal spaces, iterating the whole MNO algorithm (SteinerTC and CND solution) can introduce high computation overhead. Thus, a flexible design with low computation efforts for predicting the topologies will be much more practical in adapting to these changes in such scenarios.

7.2.2 Machine-learned Versions for QF-MAC and QF-Geo

To further approximate the optimal solution across channel scheduling and packet routing with low computation overhead, exploring machine-learned policies using only regional information can be intriguing as a practical implementation solution in real networks to achieve nearly optimal performance.

For the QF-MAC, we plan to study how to take advantage of deep learning and reinforcement learning to perform accurate interference prediction locally with lightweight sampling overhead. An embedding of states from locality (e.g., network states of neighborhood) can be learned using graph neural networks (GNNs) and then used to predict the patterns of internal and external interference finely. In order to capture the complex relationship across spatial domains (e.g., topology) and temporal domains (e.g., interference and mobility patterns), several neural network architectures (e.g., long short-term memory and graph attention network) will be involved. We will also consider techniques to deal with the jamming of the control channel itself and further explore multi-radio goodput efficiency enhancement.

For the QF-Geo, we are interested in learning-based approximations of *QF-Geo* and its refinements. So far *QF-Geo* predicts the search region of shortest paths based on a given source-destination pair and network density. Our work can be extended to a learning-based framework that considers states from a locality to predict more precise and smaller search regions to reduce search efforts. Concerning capacity-aware routing, the search region of the maximum capacity path for a given source-destination pair can vary as the interference level changes. Our focus will be on refinements of the coordination with other network layers (e.g., QF-MAC) to provide distributed multi-channel control that achieves higher capacity-delay performance,

using reinforcement learning techniques across layers. Moreover, we are interested in learning node mobility patterns to predict regional topologies and then dispatch the corresponding routing protocols to achieve high-throughput communication. Also, our work has uncovered opportunities for further enhancements of SDNs and 5G networks with a hybrid wired-wireless architecture.

7.2.3 Finer Clustering and Routing Policies for Sparse Graphs

We are motivated to learn multiple policies that respectively receive better performance under distinct sets of origin-destination pairs whose $SIM_v(m, Q^*)$ vary across nodes and graphs. According to the analysis results, using origin-destination distance, path stretch, and $SIM_v(m, Q^*)$ guides a pragmatic way for task division. The similarity of routing paths for origin-destination pairs in a graph can be classified. Then, the routing policies can be learned from different subgraphs to improve the average generalization performance in any graph.

By applying the same DNN architecture (Figure 6.1) using distance and node stretch as the input features, we additionally learn *Greedy Lax* for searching a larger routing region and propose a hybrid policy to utilize *Greedy Tensile* and *Greedy Lax* for easy instances and hard instances, respectively. Up to 7% improvement of APNSP prediction from *Greedy Tensile* is achieved using the hybrid policy in sparse graphs.

So far, the improvement in APNSP prediction accuracy using hybrid policies has been bounded by the clustering algorithm and the selected DNN models. Refining the clustering algorithm reduces the gap between the hybrid policy and the theoretical upper bound, where it is always aware of using either *Greedy Tensile* or *Greedy Lax* to achieve the best forwarder prediction at a given node and source-destination pair.

On the other hand, learning better models other than *Greedy Tensile* or *Greedy Lax* introduces a chance to increase the upper bound for a given graph.

We will extend the design of input features in our DNN architecture to have different sets of input features for different typologies. For example, we can use symbolic regression to select a set of input features and output a function of those features that have a high correlation to the optimal Q-values for a given seed graph. The choice of input is not necessarily limited to only one-hop neighbor states so as to use more local information to route much more accurately in sparse graphs. Such solutions can at least replace *Greedy Lax* for solving the hard instances. We also note that the easy and hard instances can be defined and differentiated using computational complexity. The complexity-based easy instances in APNSP can be solved by a policy using one-hop neighbor states; the complexity-based hard instances may require a model using regional or global states as the input features to make a more accurate prediction in APNSP.

Moreover, the classification of multiple instances can be learned to finely separate the set of tuples and apply corresponding routing policies for these instances to achieve near-optimal performance in sparse graphs. The hard instances can be further broken down into multiple clusters associated with corresponding routing policies. We notice that $SIM_v(m, Q^*)$ is an intuitively useful estimate to filter the minor set of instances where a single-copy policy using the features derived from one hop local information does not work well. Since $SIM_v(m, Q^*)$ compares local ranking and global ranking to show their correlation, a complete state of the entire graph is required to obtain the optimal Q-values (i.e., the length of the shortest paths). We also plan to use local

estimation to approximate the optimal Q-values so that the computation overhead can be independent of the graph size.

7.2.4 Continual Meta-learning

As the nodes or edges in a graph can change spatially or temporally, and thereby, it may yield a new topology where the applied routing policy performs not well, the model of the routing policy can be quickly refined for the new graph with limited data while still remembering the learned knowledge for previous graphs. Towards this end, a two-stage continual meta-learning approach can be studied: 1) offline meta-training to learn a meta-model that captures the inductive bias for fast learning under the guidance of the graph knowledge; 2) online continual learning to quickly adapt to new tasks based on the meta-model and also continuously refine the meta-model without forgetting.

Offline Meta-training

Suppose that we are given a set of tasks in different graphs, where each task has a dataset collected by some task policies. Since the decision-making in a fixed graph corresponds to a Markov Decision Process (MDP) problem and the graph topology is usually known to the system designer, we also consider the case where a simulator for the underlying MDP is available for task policy learning. To enable efficient learning of the meta-knowledge, one important observation here is that different tasks can have different levels of task correlations with other tasks, which also closely hinges upon the correlation among different graph structures. This consequently motivates the design of offline meta-learning, where the underlying rationale is as follows: extract

the meta-knowledge from similar tasks to enable fast adaptation and organize the meta-knowledge for dissimilar tasks in an orthogonal manner to minimize forgetting.

Online Continual Learning

Based on the learned meta-model in offline meta-training, the task policy for any new task can be quickly adapted in an online manner. Specifically, given the limited online data and graph knowledge of the new task, we can determine which subspace this new task belongs to by leveraging the learned mapping. The task policy can then be adapted from the projected meta-model onto the corresponding subspace. Clearly, the knowledge of the new task can also be utilized to further refine the meta-model. To protect the learned meta-knowledge of other clusters, we will update the meta-model only along the direction orthogonal to the subspaces of these clusters.

Moreover, while samples from nodes in a single shortest path of a single seed graph suffice for generalizable learning, in practice, learning from multiple shortest paths in one or more seed graphs may be of interest. For instance, if an ideal seed graph or shortest path is not known a priori, online learning from better or multiple candidate seed graphs and paths as they are encountered may be of interest for some applications. Along these lines, we recall that the set of ideal (and near ideal) seed graphs is relatively large in the problem we considered.

Appendix A: Approximation Ratio of SteinerTC Solution

Lemma 1. *Given a height increment δ at v, STAR-SteinerTC-ALGO returns height increments of v's logical neighbors such that the cost-to-benefit ratio of v is minimized.*

Proof. Let $incr_{OPT}$ be the increment function with height increment δ at v and the height increments of some of the logical neighbors of v. Let J be the set of chosen logical neighbors of v in $incr_{OPT}$ such that the cost-to-benefit ratio at v is optimal. Note that the benefit of $incr_{OPT}$ at v is $|J|$ that minimized the cost-to-benefit ratio. If there exists more than one vertex in J in the same component D_i, the benefit of $incr_{OPT}$ at v is less than J. Then, we can remove one vertex in $J \cap D_i$ to obtain a lower cost-to-benefit ratio at v. This contradicts the definition of $incr_{OPT}$.

Let $incr_{ALGO}$ be the increment function returned by STAR-SteinerTC-ALGO and K be the set of chosen logical neighbors of v in $incr_{ALGO}$. The benefit of $incr_{OPT}$ at v is $|K|$ since line 16 in Algorithm 1 remains only one vertex in each component. Now, we will compare the selection of J and K. First, suppose $|K|$ is equal to $|J|$. Note that, for each component having at least one logical neighbor of v, STAR-SteinerTC-ALGO only remains one vertex with the lowest cost increment. $incr_{ALGO}$ also considers the same set of vertices, denoted as L, because choosing another vertex in the component makes higher cost increment but identical benefit. Since the for loop from lines 19

to 24 chooses the first $|K|$ elements in L with ascending order of cost increment, the sum of cost increments with size $|K|$ is minimized. Thus, $incr_{ALGO}$, equal to $incr_{OPT}$, is optimal.

Second, suppose $|K|$ is less than $|J|$. Since K is the first $|K|$ elements in L and K is the set with minimal cost increment in size $|K|$, $J \cap K = K$ and $J \setminus K$ contains at least one vertex u not in the components of vertices in K. Note that vertex u is the one with minimal cost increment in its component. Thus, u is in L. However, the for loop from lines 19 to 24 in Algorithm 1 considers all sizes from 1 to $|L|$ to determine the lowest cost-to-benefit ratio. u is selected in $incr_{ALGO}$ if and only if u and the vertices with lower cost increments than u in L make lowest cost-to-benefit ratio. This conflicts with the assumption. Hence, $|K|$ must be identical to $|J|$. According to the first statement, $incr_{ALGO}$ is optimal.

Finally, suppose $|K|$ is larger than $|J|$. Since $J \cap K = J$, we can obtain a lower cost-to-benefit ratio by removing $L[|J| + 1], \ldots, L[|K|]$ from K. However, the set $L[1 \ldots |J|]$ is considered by STAR-SteinerTC-ALGO and chosen as $incr_{ALGO}$ if and only if $L[1 \ldots |J|]$ makes lowest cost-to-benefit ratio. The result conflicted with the assumption. Hence, $|K|$ must be identical to $|J|$ and thus $incr_{ALGO}$ is optimal. □

Lemma 2. *The cost-to-benefit ratio of the height increment selected in each iteration of SteinerTC-ALGO is at most twice the cost-to-benefit ratio of any height increment centered at v.*

Proof. Since SteinerTC-ALGO remains the same as TC-ALGO and STAR-SteinerTC-ALGO returns local optimum, it also achieves a 2-approximation of the optimal height increment centered at v as TC-ALGO does, which is proved in [61]. □

Definition 2. *A spider is a tree containing at least two leaves and having disjoint paths from the root to all leaves. There is, at most, one vertex with a degree larger than 2 in a spider. A root of a spider is a vertex with edge-disjoint paths to all leaves of the spider. The root is unique if and only if a spider contains more than two leaves.*

Definition 3. *Let $G = (V, E)$ be an undirected connected graph, and let S be a subset of V. A spider decomposition $SD(G, S)$ is a set of vertex-disjoint spiders in G such that the union of leaves of spiders in $SD(G, S)$ contains all vertices in S.*

Lemma 3. *[40]. Let $G = (V, E)$ be an undirected connected graph and let S be a subset of V, where $|S| \geq 2$. There exists a spider decomposition $SD(G, S)$ in G.*

Lemma 4. *Consider iteration i in the greedy algorithm SteinerTC-ALGO. Let the height function at the beginning of iteration i be h_{i-1}. Let OPT be the cost of optimal height function h_{OPT}. The ratio of $\frac{cost}{benefit-1}$ associated with the height increment selected by iteration i is at most $\frac{2OPT}{|cover(h_{i-1})|}$.*

Proof. Let c_i be the cost increment and b_i be the benefit associated with the height increment determined in iteration i. Note that edges in $cover(h_{OPT})$ associated with the corresponding height increments can be used to connect $|cover(h_{i-1})|$ components into a tree. Let $cover(h_{i-1})$ have ϕ_{i-1} components $D_1, \ldots, D_{\phi_{i-1}}$. Let T_i be the graph obtained from $cover(h_{OPT})$ by contracting $D_j \cap cover(h_{OPT}), j = 1, \ldots, \phi_{i-1}$ to a supervertex. That is to say, starting with $cover(h_{OPT})$, for each component D_j, $D_j \cap cover(h_{OPT})$ is merged into a supervertex. All the supervertices will be connected in one single component after the final iteration is done. Note that the vertices in $D_j \setminus cover(h_{OPT})$, that are non-terminals, have already connected by the supervertex

contracted from $D_j \cap cover(h_{OPT})$. Connecting all supervertices yields a Steiner tree spanning all terminals.

Let S denote the set of the ϕ_{i-1} supervertices. According to Lemma 3, T_i contains a spider decomposition $SD(T_i, S)$. Each spider in $SD(T_i, S)$ is associated with height increments (derived from h_{OPT}) at its leaves (i.e., supervertices). The cost increment from $cover(h_{i-1})$ to T_i is at most OPT. Let $r_1, \ldots, r_w$ be the roots of the spiders in $SD(T_i, S)$ and $m_1, \ldots, m_w$ be the number of supervertices in S in each of these spiders. Let sc_j denote the cost increment to form the spider with root r_j, where $m_j \geq 2$. Note that a spider with root r_j spans a subset of components $D_1, \ldots, D_{\phi_{i-1}}$, which is the m_j components corresponding to m_j supervertices in this spider. Accordingly, the cost-to-benefit ratio at r_j is $\frac{sc_j}{m_j - 1}$. Note also that the cost-to-benefit ratio at r_j, a non-terminal with fixed height, can be considered the cost-to-benefit ratio at a terminal v in the supervertices of the same spider, that is induced by an edge (v, u), $u \in B$, in the spider associated with the height increment $h^+(v)$. Therefore, the cost-to-benefit ratio at v in the spider is at least the cost-to-benefit ratio determined by STAR-SteinerTC-ALGO in consideration of the same terminal v and height increment $h^+(v)$.

Since SteinerTC-ALGO chooses a vertex with lowest cost-to-benefit ratio in each iteration, for each spider in $SD(T_i, S)$, $\frac{c_i}{b_i} \leq \frac{sc_j}{m_j - 1}, j = 1, \ldots, w$. Moreover, $\frac{c_i}{b_i}$ is at most the average cost-to-benefit ratio over spiders in $SD(T_i, S)$, thus we have:

$$\begin{aligned} & \frac{c_i}{b_i} \leq \min_j \frac{sc_j}{m_j - 1} \leq \frac{\sum_{j=1}^{w} sc_j}{\sum_{j=1}^{w} (m_j - 1)} \\ \Longrightarrow \; & \frac{c_i}{b_i + 1} \leq \min_j \frac{sc_j}{m_j} \leq \frac{\sum_{j=1}^{w} sc_j}{\sum_{j=1}^{w} m_j} \\ \Longrightarrow \; & \frac{c_i}{b_i + 1} \sum_{j=1}^{w} m_j \leq \sum_{j=1}^{w} sc_j \leq OPT. \end{aligned}$$

Due to $\sum_{j=1}^{w} m_j = \phi_{i-1}$,

$$\frac{c_i}{b_i + 1} \leq \frac{OPT}{\phi_{i-1}} = \frac{OPT}{|cover(h_{i-1})|}. \tag{A.1}$$

Note that $c_i \leq sc_j$. According to Lemma 2, a single iteration of the greedy algorithm SteinerTC-ALGO finds a 2-approximation of the optimal height increment, the minimum ratio of $\frac{cost}{benefit-1}$ for the height increment selected by SteinerTC-ALGO is thus at most $2 \times \frac{OPT}{|cover(h_{i-1})|} = \frac{2OPT}{|cover(h_{i-1})|}$. □

From Lemma 4, the approximation ratio of $2\ln|A|OPT$ in Theorem 2 is shown as follows. According to the definition of benefit, $\phi_i = \phi_{i-1} - b_i$ in iteration i. Substituting Eq. A.1 into this equation, we obtain

$$\phi_i = \phi_{i-1} - b_i \leq \phi_{i-1} - \frac{b_i+1}{2} \leq \phi_{i-1}(1 - \frac{c_i}{2OPT}).$$

Let the total number of iterations taken in SteinerTC-ALGO be p and $\phi_p = 1$. We have

$$\begin{array}{r} \phi_p = \phi_0 \prod_{j=1}^{p} (1 - \frac{c_j}{2OPT}) \\ \Longrightarrow\ 0 = \ln \frac{\phi_0}{\phi_p} + \ln \prod_{j=1}^{p} (1 - \frac{c_j}{2OPT}) \end{array}$$

By using $\ln(1+x) \leq \ln x$, we get

$$\begin{array}{r} 0 = \ln \frac{\phi_0}{\phi_p} + \ln \prod_{j=1}^{p} (1 - \frac{c_j}{2OPT}) \\ \leq \ln \frac{\phi_0}{\phi_p} + \sum_{j=1}^{p} \frac{-c_j}{2OPT} \\ \Longrightarrow\ \frac{1}{2OPT} \sum_{j=1}^{p} c_j \leq \ln(\frac{\phi_0}{\phi_p}) \\ \Longrightarrow\ \sum_{j=1}^{p} c_j \leq 2\ln|A|OPT \end{array}$$

This proves that the total cost for the height function h returned by SteinerTC-ALGO is at most $2\ln|A|OPT$

Appendix B: Analysis of Configuration Space and Performance in Channel Hopping Sequences

This section analyzes the configuration space of the network and scheduling parameters and compares the performance across different scheduling of channel hopping sequences. The objective of the analysis is to understand the vulnerability of TSCH and explore the feasibility of other scheduling schemes to improve the average throughput in consideration of channel collision across radios associated with the same or different flows within an interference region. Again, we note that, in an interference region, the Euclidean distance between any two radios is not greater than both the interference radius R_i and the transmission radius R_t. Therefore, any two radios can interfere with each other during data transmission.

B.1 Configuration Space of Channel Hopping Scheduling

We let U denote the number of all channels, and M denote the number of radios in a given interference region. Note that $U \geq M$ in Eq. 4.3 so that all radios can use the disjoint channel at a slot. F active flows deliver data packets within this range at each slot. Since the network enables concurrent transmissions, each radio associated with a flow sends packets at each slot. Without loss of generality, we let $K = \frac{M}{F}$ represent the number of radios in a flow within the interference region. L defines the

length of a cyclic sequence of frequencies. Each frequency disjoints in a sequence if $L \leq U$. When $L > U$, the sequence is broken down into $\lceil \frac{L}{U} \rceil$ segments, where up to U frequencies are disjoint.

B.2 Scheduling of Channel Hopping Sequences

Random channel hopping sequence. Each radio locally generates a cyclic sequence of frequencies without coordination. Channel collision can happen for any two radios associated with either the same or different flows.

Global channel hopping sequence. TSCH applies a globally fixed cyclic sequence of frequencies shared by all radios. It relies on channel offset schedules to avoid collision among transmitting radios within an interference region.

Fixed per-flow channel hopping sequence. All radios associated with the same flow share a cyclic sequence of frequencies. The sequence is randomly chosen from a set of channels at a flow setup. Within each flow, the channel offset of the next forwarder is right-shifted by one to avoid intra-flow channel collision.

B.3 Analysis of Collisions per Slot

In order to understand the survivability of sequence schedulers in a network without additional control overhead across data flows, all sequence schedulers are performed without inter-flow coordination. Thus, inter-flow channel collision may occur because a radio is unaware of the radio's sequence information of other flows. On the other hand, intra-flow channel collision can be avoided by simply shifting the channel offset by one position. This offset shifting can be done at a flow setup, which is to

select a routing path and assign the transmission/reception channel sequences for a flow.

In a scheduler with random sequences, the sequence assignment at each radio is independent. Thus, both intra-flow and inter-flow channel collisions may occur. At radio v, the probability of not getting collided with one other radio u at a slot is $1 - \frac{1}{U}$. Since there are $M - 1$ radios potentially interfering with the channel used by radio v, the probability of collision per slot at a radio is calculated as follows:

$$P_{random} = 1 - (\frac{U-1}{U})^{M-1} \tag{B.1}$$

In a global channel hopping sequence scheduler, the offset is randomly chosen at the source node at the flow setup. Then, the offset is added by one whenever the flow control message is propagated to the next hop in a downstream direction. For each flow with radios $v_1, ..., v_K$, there are K disjoint channels selected for $v_1, ..., v_K$ at a slot. Given a selected channel associated with radio $v_{x_1} \in$ flow f_x, the probability that all radios associated with other flow f_y cause no conflict is $\frac{L-K}{L}$. Since there are $F - 1$ flows potentially interfering with the channel used by radio v_{x_1}, the probability of collision per slot at a radio is calculated as follows:

$$P_{global} = 1 - (\frac{L-K}{L})^{F-1} \tag{B.2}$$

In the scheduler with a fixed per-flow channel hopping sequence, similar to the global channel hopping sequence, the offset is assigned within a flow in a downstream direction to avoid intra-flow channel collision. However, compared to a global channel hopping sequence with only L channels used in the interference range, the scheduler allows each flow randomly to select L channels from the set of all channels. Therefore,

given a selected channel associated with radio $v_{x_1} \in$ flow f_x, the probability that all radios associated with other flow f_y cause no conflict is $\frac{U-K}{U}$. Since there are $F-1$ flows potentially interfering with the channel used by radio v_{x_1}, the probability of collision per slot at a radio is calculated as follows:

$$P_{perflow} = 1 - (\frac{U-K}{U})^{F-1} \tag{B.3}$$

Note that $P_{perflow}$ is independent of L. Thus, only a short sequence is needed to schedule fixed per-flow channel hopping sequences. This feature improves the flexibility and diversity of chosen channels in each flow and thus saves the payload size in control messages of flow setup.

Lemma 5. *In an interference region with concurrent flows, where $U \geq M \geq F > 1$ and $\frac{M}{F} = K \geq 1$, scheduling fixed per-flow channel hopping sequences achieves a performance not worse than the one in scheduling global or random channel hopping sequences: $P_{perflow} \leq P_{random}$ and $P_{perflow} \leq P_{global}$.*

Proof. In the comparison of Eq. B.2 and Eq. B.3, $P_{perflow}$ is not greater than P_{global} because of $U \geq L$ and $\frac{U-K}{U} \geq \frac{L-K}{L}$. Namely, scheduling fixed per-flow channel hopping sequences yields an average throughput that is not worse than that of a global channel hopping sequence within an interference region.

In the comparison of Eq. B.1 and Eq. B.3, to show that $(\frac{U-1}{U})^{M-1} \leq (\frac{U-K}{U})^{F-1}$,we consider three cases: (i) $U \geq M = F$, (ii) $U = M \geq F$, (iii) $U > M > F$.

Case i: $U \geq M = F, K = 1$. we have $(\frac{U-1}{U})^{M-1} = (\frac{U-K}{U})^{F-1}$

Case ii: $U = M \geq F$. we have$(\frac{U-1}{U})^{M-1} = (\frac{U-1}{U})^{U-1}$ and $(\frac{U-K}{U})^{F-1} = (\frac{U-\frac{U}{F}}{U})^{F-1} = (\frac{F-1}{F})^{F-1}$. Because $(\frac{x-1}{x})^{x-1}$ is monotonically decreasing when $x \geq 1$, $(\frac{U-1}{U})^{U-1} < (\frac{F-1}{F})^{F-1}$ holds if $U > F \geq 1$. Therefore, we have $(\frac{U-1}{U})^{M-1} < (\frac{U-K}{U})^{F-1}$.

Case iii: $U > M > F$. We first show that the inequality of $(\frac{U-1}{U})^{M-1} < (\frac{U-K}{U})^{F-1}$ can be derived to the following inequality:

$$\begin{aligned} & (\tfrac{U-1}{U})^{M-1} < (\tfrac{U-K}{U})^{F-1} \\ \iff & (1-\tfrac{1}{U})^{\frac{M-1}{F-1}} < 1-\tfrac{K}{U} = 1-\tfrac{M}{UF} \end{aligned} \tag{B.4}$$

$$\begin{aligned} & 0 < (\tfrac{U-1}{U})^{M-1} < (\tfrac{U-K}{U})^{F-1} < 1 \\ \iff & \log(\tfrac{U-1}{U})^{M-1} < \log(\tfrac{U-K}{U})^{F-1} \\ \iff & (M-1)(\log(U-1)-\log U) < \\ & (F-1)(\log(U-K)-\log U) \\ \iff & \tfrac{M-1}{F-1}(\log(U-1)-\log U) < log(U-K)-\log U \\ \iff & \tfrac{M-1}{F-1} > \tfrac{\log(U-K)-\log U}{\log(U-1)-\log U} \\ \iff & \tfrac{M-1}{F-1} > \tfrac{\log U-\log(U-K)}{\log U-\log(U-1)} \\ \iff & \tfrac{M-1}{F-1} > \tfrac{\log U-(\log U+\log(1-\frac{K}{U}))}{\log U-(\log U+\log(1-\frac{1}{U}))} \\ \iff & \tfrac{M-1}{F-1} > \tfrac{\log(1-\frac{K}{U})}{\log(1-\frac{1}{U})} \\ \iff & \tfrac{M-1}{F-1}\log(1-\tfrac{1}{U}) < \log(1-\tfrac{K}{U}) \\ \iff & (1-\tfrac{1}{U})^{\frac{M-1}{F-1}} < 1-\tfrac{K}{U} = 1-\tfrac{M}{UF} \end{aligned}$$

Note that the exponential function of $f(M) = (1-\frac{1}{U})^{\frac{M-1}{F-1}}$ is monotonically decreasing when $U \geq M \geq F$ and the exponential function of $f(M) = (1-\frac{1}{U})^{\frac{M-1}{F-1}}$ and the linear function of $g(M) = 1-\frac{M}{UF}$ has at most two intersections, where one of them occurs at $M = F$. Also, we know that $(\frac{U-1}{U})^{M-1} < (\frac{U-\frac{M}{F}}{U})^{F-1}$ and $(1-\frac{1}{U})^{\frac{M-1}{F-1}} < 1-\frac{M}{UF}$ hold at $U = M$, according to the derivation in Case ii and Eq. B.4. Based on the two statements that $(1-\frac{1}{U})^{\frac{M-1}{F-1}} < 1-\frac{M}{UF}$ at $U = M$ and $(1-\frac{1}{U})^{\frac{M-1}{F-1}} = 1-\frac{M}{UF}$ at $M = F$, we can derive that $(1-\frac{1}{U})^{\frac{M-1}{F-1}} < 1-\frac{M}{UF}$, for $M \in (F, U]$. This is because that $f(M) = (1-\frac{1}{U})^{\frac{M-1}{F-1}}$ is monotonically decreasing and the possibly remaining one intersection of $f(M)$ and $g(M)$ will not occur in the range of (F, U). (Otherwise, $(1-\frac{1}{U})^{\frac{M-1}{F-1}} > 1-\frac{M}{UF}$.) Hence, $(\frac{U-1}{U})^{M-1} < (\frac{U-K}{U})^{F-1}$, for $U > M > F$, is proved.

According to the three cases above, $(\frac{U-1}{U})^{M-1} \leq (\frac{U-K}{U})^{F-1}$, for $U \geq M \geq F > 1$, and then $P_{perflow} \leq P_{random}$ is proved.

□

Bibliography

[1] Microsoft airband initiative.

[2] IEEE standard for low-rate wireless networks. *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pages 1–709, 2016.

[3] ISA100, 2021.

[4] ns-3 Direct Code Execution (DCE), 2021.

[5] WirelessHART, 2021.

[6] Nicola Accettura, Maria Rita Palattella, Gennaro Boggia, Luigi Alfredo Grieco, and Mischa Dohler. Decentralized traffic aware scheduling for multi-hop low power lossy networks in the internet of things. In *2013 IEEE 14th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–6. IEEE, 2013.

[7] Rodrigo Aldecoa, Chiara Orsini, and Dmitri Krioukov. Hyperbolic graph generator. *Computer Physics Communications*, 196:492–496, 2015.

[8] Eiman Alotaibi and Biswanath Mukherjee. A survey on routing algorithms for wireless ad-hoc and mesh networks. *Computer Networks*, 56(2):940–965, 2012.

[9] Bassam Aoun, Raouf Boutaba, Youssef Iraqi, and Gary Kenward. Gateway placement optimization in wireless mesh networks with qos constraints. *IEEE Journal on Selected Areas in Communications*, 24(11):2127–2136, 2006.

[10] Michael Baddeley, Adnan Aijaz, Usman Raza, Aleksandar Stanoev, Yichao Jin, Markus Schuß, Carlo Alberto Boano, and George Oikonomou. 6TiSCH++ with Bluetooth 5 and concurrent transmissions. *arXiv preprint arXiv:2010.09529*, 2020.

[11] Jyotirmoy Banik, Ricardo Arjona, Marco Tacca, Miguel Razo, Andrea Fumagalli, Kumaran Vijayasankar, and Arvind Kandhalu. Improving performance in industrial Internet of Things using multi-radio nodes and multiple gateways. In *2017 International Conference on Computing, Networking and Communications (ICNC)*, pages 604–608. IEEE, 2017.

[12] Jieyi Bi, Yining Ma, Jiahai Wang, Zhiguang Cao, Jinbiao Chen, Yuan Sun, and Yeow Meng Chee. Learning generalizable models for vehicle routing problems via knowledge distillation. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[13] Lynne Billard and Edwin Diday. Symbolic regression analysis. In *Classification, clustering, and data analysis: recent advances and applications*, pages 281–288. Springer, 2002.

[14] Marián Boguná, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the internet with hyperbolic mapping. *Nature Communications*, 1(1):62, 2010.

[15] Azzedine Boukerche, Begumhan Turgut, Nevin Aydin, Mohammad Z Ahmad, Ladislau Bölöni, and Damla Turgut. Routing protocols in ad hoc networks: A survey. *Computer Networks*, 55(13):3032–3080, 2011.

[16] Fraser Cadger, Kevin Curran, Jose Santos, and Sandra Moffett. A survey of geographical routing in wireless ad-hoc networks. *IEEE Communications Surveys & Tutorials*, 15(2):621–653, 2012.

[17] Chih-Min Chao, Chia-Tsun Chen, and Hsin-Chung Huang. An adjustable channel hopping algorithm for multi-radio cognitive radio networks. *Computer Networks*, 170:107107, 2020.

[18] Kameswari Chebrolu, Bhaskaran Raman, and Sayandeep Sen. Long-distance 802.11 b links: performance measurements and experience. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, pages 74–85, 2006.

[19] Yung-Fu Chen, Kenneth W Parker, and Anish Arora. QF-Geo: Capacity aware geographic routing using bounded regions of wireless meshes. *arXiv preprint arXiv:2305.05718*, 2023.

[20] Xia Cheng, Junyang Shi, and Mo Sha. Cracking the channel hopping sequences in IEEE 802.15. 4e-based industrial TSCH networks. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, pages 130–141, 2019.

[21] Xia Cheng, Junyang Shi, Mo Sha, and Linke Guo. Launching smart selective jamming attacks in WirelessHART networks. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[22] Reuven Cohen and Liran Katzir. The generalized maximum coverage problem. *Information Processing Letters*, 108(1):15–22, 2008.

[23] Carlos Cordeiro, Kiran Challapali, Dagnachew Birru, and Sai Shankar. Ieee 802.22: the first worldwide wireless standard based on cognitive radios. In *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.*, pages 328–337. IEEE, 2005.

[24] Diego Dujovne, Thomas Watteyne, Xavier Vilajosana, and Pascal Thubert. 6tisch: deterministic ip-enabled industrial internet (of things). *IEEE Communications Magazine*, 52(12):36–41, 2014.

[25] Facebook. Terragraph, 2019.

[26] Gregory G Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical report, University of Southern California Marina Del Rey Information Sciences Inst, 1987.

[27] Roland Flury, Sriram V Pemmaraju, and Roger Wattenhofer. Greedy routing with bounded stretch. In *IEEE INFOCOM*, pages 1737–1745, 2009.

[28] E Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[29] Matthias Grossglauser and David NC Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(4):477–486, 2002.

[30] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[31] Thomas R Halford and Keith M Chugg. Barrage relay networks. In *2010 Information Theory and Applications Workshop (ITA)*, pages 1–8. IEEE, 2010.

[32] Refael Hassin, R Ravi, and F Sibel Salman. Approximation algorithms for a capacitated network design problem. *Algorithmica*, 38(3):417–431, 2004.

[33] Ramin Hekmat and Piet Van Mieghem. Interference in wireless multi-hop ad-hoc networks and its effect on network capacity. *Wireless Networks*, 10:389–399, 2004.

[34] Rodrigo Teles Hermeto, Antoine Gallais, and Fabrice Theoleyre. Scheduling for IEEE802. 15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey. *Computer Communications*, 114:84–105, 2017.

[35] Ting-Chao Hou and Victor Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, 34(1):38–44, 1986.

[36] Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem. *arXiv preprint arXiv:2110.05291*, 2021.

[37] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[38] Brad Karp and Hsiang-Tsung Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 243–254, 2000.

[39] Junseok Kim and Younggoo Kwon. Interference-aware geographical routing for sensor-nets in indoor environments. In *2009 International Conference on Information Networking*, pages 1–5. IEEE, 2009.

[40] Philip N. Klein and R Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. Algorithms*, 19(1):104–115, 1995.

[41] Jon M Kleinberg. Navigation in a small world. *Nature*, 406(6798):845–845, 2000.

[42] Y-B Ko and Nitin H Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 101–110. IEEE, 1999.

[43] Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of Economic Perspectives*, 15(4):143–156, 2001.

[44] Evangelos Kranakis. Compass routing on geometric networks. In *Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG 1999), Vancouver, August*, 1999.

[45] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. An algorithmic approach to geographic routing in ad hoc and sensor networks. *IEEE/ACM Transactions On Networking*, 16(1):51–62, 2008.

[46] Jing Li, Wenjie Zeng, and Anish Arora. Chameleon: On the energy efficiency of exploiting multiple frequencies in wireless sensor networks. In *International Conference on Broadband Communications, Networks and Systems*, pages 138–157. Springer, 2010.

[47] Ting-Yu Lin, Kun-Ru Wu, and Guang-Chuen Yin. Channel-hopping scheme and channel-diverse routing in static multi-radio multi-hop wireless networks. *IEEE Transactions on Computers*, 64(1):71–86, 2013.

[48] Tzu-Hsiang Lin, Guu-Chang Yang, and Wing C Kwong. A homogeneous multi-radio rendezvous algorithm for cognitive radio networks. *IEEE Communications Letters*, 23(4):736–739, 2019.

[49] Victoria Manfredi, Alicia Wolfe, Xiaolan Zhang, and Bing Wang. Learning an adaptive forwarding strategy for mobile wireless networks: Resource usage vs. latency. In *Reinforcement Learning for Real Life (RL4RealLife) Workshop in the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[50] Victoria Manfredi, Alicia P Wolfe, Bing Wang, and Xiaolan Zhang. Relational deep reinforcement learning for routing in wireless networks. In *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 159–168, 2021.

[51] KW Matthee, Gregory Mweemba, Adrian V Pais, Gertjan Van Stam, and Marijn Rijken. Bringing internet connectivity to rural zambia using a collaborative approach. In *2007 International Conference on Information and Communication Technologies and Development*, pages 1–12. IEEE, 2007.

[52] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[53] Rafael Molina-Masegosa and Javier Gozalvez. LTE-V for Sidelink 5G V2X vehicular communications: A new 5G technology for short-range vehicle-to-everything communications. *IEEE Vehicular Tech. Magazine*, 12(4):30–39, 2017.

[54] SH Muggleton. Hypothesizing an algorithm from one example: the role of specificity. *Philosophical Transactions of the Royal Society A*, 381(2251):20220046, 2023.

[55] Stephen Muggleton. Learning from positive data. In *International Conference on Inductive Logic Programming*, pages 358–376. Springer, 1996.

[56] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[57] Randolph Nelson and Leonard Kleinrock. The spatial capacity of a slotted aloha multihop packet radio network with capture. *IEEE Transactions on Communications*, 32(6):684–694, 1984.

[58] Yong Niu, Yong Li, Depeng Jin, Li Su, and Athanasios V Vasilakos. A survey of millimeter wave communications (mmwave) for 5G: Opportunities and challenges. *Wireless Networks*, 21(8):2657–2676, 2015.

[59] Charalampos Orfanidis, Atis Elsts, Paul Pop, and Xenofon Fafoutis. TSCH evaluation under heterogeneous mobile scenarios. *IoT*, 2(4):656–668, 2021.

[60] Maria Rita Palattella, Nicola Accettura, Mischa Dohler, Luigi Alfredo Grieco, and Gennaro Boggia. Traffic aware scheduling algorithm for reliable low-power multi-hop ieee 802.15. 4e networks. In *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications-(PIMRC)*, pages 327–332. IEEE, 2012.

[61] Debmalya Panigrahi, Partha Dutta, Sharad Jaiswal, KVM Naidu, and Rajeev Rastogi. Minimum cost topology construction for rural wireless mesh networks. In *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, pages 771–779. IEEE, 2008.

[62] Fragkiskos Papadopoulos, Dmitri Krioukov, Marián Boguná, and Amin Vahdat. Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. In *IEEE INFOCOM*, pages 1–9, 2010.

[63] Bhaskaran Raman and Kameswari Chebrolu. Design and evaluation of a new mac protocol for long-distance 802.11 mesh networks. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, pages 156–169, 2005.

[64] Joao Reis, Miguel Rocha, Truong Khoa Phan, David Griffin, Franck Le, and Miguel Rio. Deep neural networks for network routing. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[65] F Sibel Salman, Joseph Cheriyan, Ramamoorthi Ravi, and Sairam Subramanian. Approximating the single-sink link-installation problem in network design. *SIAM Journal on Optimization*, 11(3):595–610, 2001.

[66] Science and Technology Directorate Report. TSM tactical radios for urban and subterranean environments, March 2018.

[67] Sayandeep Sen and Bhaskaran Raman. Long distance wireless mesh network planning: problem formulation and solution. In *Proceedings of the 16th international conference on World Wide Web*, pages 893–902. ACM, 2007.

[68] Joseph Sill. Monotonic networks. *Advances in neural information processing systems*, 10, 1997.

[69] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. A survey on LPWA technology: LoRa and NB-IoT. *ICT Express*, 3(1):14–21, 2017.

[70] Ridha Soua, Pascale Minet, and Erwan Livolant. Modesa: An optimized multichannel slot assignment for raw data convergecast in wireless sensor networks. In *2012 IEEE 31st international performance computing and communications conference (IPCCC)*, pages 91–100. IEEE, 2012.

[71] Ridha Soua, Pascale Minet, and Erwan Livolant. Wave: a distributed scheduling algorithm for convergecast in ieee 802.15. 4e TSCH networks. *Transactions on Emerging Telecommunications Technologies*, 27(4):557–575, 2016.

[72] Carl R Stevenson, Gerald Chouinard, Zhongding Lei, Wendong Hu, Stephen J Shellhammer, and Winston Caldwell. Ieee 802.22: The first cognitive radio wireless regional area network standard. *IEEE Communications magazine*, 47(1):130–138, 2009.

[73] Hideaki Takagi and Leonard Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

[74] Hiromitsu Takahashi. An approximate solution for the steiner problem in graphs. *Math. Japonica.*, 6:573–577, 1990.

[75] Guang Tan, Marin Bertier, and A-M Kermarrec. Convex partition of sensor networks and its use in virtual coordinate geographic routing. In *IEEE INFOCOM*, pages 1746–1754, 2009.

[76] Guang Tan, Marin Bertier, and Anne-Marie Kermarrec. Visibility-graph-based shortest-path geographic routing in sensor networks. In *IEEE INFOCOM*, pages 1719–1727, 2009.

[77] Guang Tan and Anne-Marie Kermarrec. Greedy geographic routing in large-scale sensor networks: A minimum network decomposition approach. *IEEE/ACM Transactions on Networking*, 20(3):864–877, 2011.

[78] Xuesong Jonathan Tan, Jieran Wang, and Yongjun Yuan. Difference-set-based channel hopping for minimum-delay blind rendezvous in multi-radio cognitive radio networks. *IEEE Transactions on Vehicular Technology*, 68(5):4918–4932, 2019.

[79] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[80] Kevin Verbeek and Subhash Suri. Metric embedding, hyperbolic space, and social networks. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 501–510, 2014.

[81] Pascal Von Rickenbach, Stefan Schmid, Roger Wattenhofer, and Aaron Zollinger. A robust interference model for wireless ad-hoc networks. In *19th IEEE International Parallel and Distributed Processing Symposium*, pages 8–pp. IEEE, 2005.

[82] Frederick W. Vook, Amitava Ghosh, Emilio Diarte, and Michael Murphy. 5G New Radio: Overview and Performance. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 1247–1251, 2018.

[83] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[84] Myounggyu Won and Radu Stoleru. A low-stretch-guaranteed and lightweight geographic routing protocol for large-scale wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 11(1):1–22, 2014.

[85] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):5057–5069, 2021.

[86] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. NeuroLKH: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 7472–7483, 2021.

[87] Feng Xue and Panganamala R Kumar. *Scaling laws for ad hoc wireless networks: an information theoretic approach*. Now Publishers Inc, 2006.

[88] Bo Yang, Wei Liang, Meng Zheng, and Ying-Chang Liang. Fully distributed channel-hopping algorithms for rendezvous setup in cognitive multiradio networks. *IEEE Transactions on Vehicular Technology*, 65(10):8629–8643, 2015.