20011372

Shanghai Jiao Tong University
University of Michigan – Shanghai Jiao Tong University Joint Institute

Boost Throughput of Wireless Networks via AI-Assisted Algorithms at Different Architectural Levels

by

Tianxin Wang

A thesis submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy in
Information and Communication Engineering at Shanghai Jiao Tong University

Committee in charge:
Prof. Xudong Wang, Chair
Prof. Geoffrey Ye Li
Prof. Meixia Tao
Prof. Chong Han
Prof. Yibo Pi

Shanghai
September 2025

20011372

20011372

上海交通大学
密西根学院

基于AI算法在不同架构层提升无线网络吞吐量的研究

王天欣

上海交通大学密西根学院博士学位论文
信息与通信工程专业

委员会： 上海
王旭东教授，主席 **2025年9月**
李烨教授
陶梅霞教授
韩充教授
皮宜博教授

20001372

20011372

Abstract

Next-generation wireless networks are expected to achieve extremely high throughput, supporting diverse bandwidth-intensive applications such as Meta-verse and digital twins. At the same time, artificial intelligence (AI), especially deep learning methods, is going to play an indispensable role in future network designs. Given the above urgent needs of pursuing high throughput and integrating AI into wireless networks, this dissertation is dedicated to boosting network throughput with AI-assisted algorithms. The overall design framework consists of three architectural levels, namely network infrastructure level, resource management level, and link level. At each level, critical research problems are stated and AI-based approaches are designed to resolve them.

At the network infrastructure level, as deployment of radio sites becomes significantly denser, wireless backhauling provides a more cost-effective and flexible solution than fiber-based wired backhauling. With wireless backhauling, building high-throughput backhaul mesh networks is one of the most intriguing problems to be studied, which is critical for ensuring broad coverage and connectivity of future networks. To maintain high throughput, the architecture of a backhaul mesh network needs to be augmented by adding relays. However, how to place relays at appropriate sites poses two challenges: 1) there lacks a theory to capture the relationship between a certain change of network architecture and its throughput gain; 2) selecting the best sites for relays is a complicated combinatorial problem. To tackle the first challenge, a clique-based bottleneck theory is first established, through which a clique-based bottleneck structure of a given network architecture is constructed to determine the network throughput. Based on this bottleneck structure, clique gradients are then computed to quantify the impact of each clique on the overall network throughput. With the clique-based bottleneck theory, the second challenge is resolved by embedding clique gradients into a deep reinforcement learning (DRL) scheme. Specifically, the DRL actions are masked such that only the relay sites that match the highest clique gradients are selected. This DRL-based relay placement (DeepRP) scheme is evaluated via extensive simulations, and performance results show that it can boost network throughput by more than 50%, which is 10.4–32.1% higher than those of baseline schemes.

At the resource management level, network slicing is considered as an indispensable technology for managing resources for diverse network services within a shared physical network infrastructure. To realize network slicing for radio access networks, one of the most intriguing tasks is slice enforcement over air interfaces across multiple cells. The challenges lie in several aspects. First, resources allocated to different slices must achieve soft isolation at the link level. Second, users' diverse quality-of-service (QoS) requirements must be satisfied even when communication links experience fading and interference. Third, long-term slicing policies must be conformed, no matter how unbalanced they are. To address these challenges, link-level slice enforcement is first formulated as a resource allocation problem that minimizes radio resource consumption while ensuring link-level soft slice isolation, guaranteeing users' diverse QoS requirements, and conforming to slicing policies. Next, this problem is tackled via a DRL-based approach, through which LinkSlice is designed as an iterative two-stage algorithm. The first stage determines transmission rates for each link based on DRL. It is embedded with a graph neural network (GNN) to characterize link interference. Based on the transmission rates from the first stage, the second stage allocates resources to each slice. Performance results show that LinkSlice converges quickly to a near-optimal solution. It gracefully tackles the three challenges of link-level slice enforcement while further improving throughput by 18.5%.

At the link level, designing neural-network-based wireless receivers (i.e., neural receiver) is one of the most promising technologies in AI-native communications. An uplink neural receiver is usually pretrained offline and

20001372

deployed online at a base station (BS). However, its performance can degrade in varying channel environments, so the neural receiver must be retrained online to maintain high performance. Two aspects of online learning are considered: 1) how to realize short-term fast adaptation; 2) how to improve long-term online generalization ability of one neural receiver. Targeted at these two aspects, a short-term online adaptation scheme of a partially-learned neural receiver (i.e., neural channel estimator) is first designed, and then a long-term collaborative online training framework of multiple fully-learned neural receivers is developed, which are elaborated as follows.

First, short-term online adaptation of a neural channel estimator is considered. Conventional methods demand ground-truth channel coefficients as supervised labels, but such labels are unavailable online. To this end, a self-supervised task is introduced on top of the original channel-estimation task to facilitate label-free adaptation of neural estimators. Specifically, this task randomly masks a fraction of resource elements in each received frame and reconstructs such masked parts. To enable effective reconstruction, the task input must incorporate two components: the unmasked parts and estimated data-symbols of masked parts. These estimated symbols are obtained via an online symbol-recovery mechanism, so no additional pilot overhead is incurred. To consolidate the self-supervised task with the original task, a two-branch masked auto-encoder (MAE) model called ChannelMAE is developed, with each branch dedicated to one task. The two branches share the same encoder but use separate decoders. During online adaptation, the encoder is updated by optimizing the self-supervised branch, which learns channel statistical features and shares them with the channel-estimation branch. Therefore, online channel-estimation accuracy is much improved. Extensive experiments show that ChannelMAE reduces channel-estimation error by up to 71.8% and 87.1% compared with the pretrained model and the state-of-the-art adaptation scheme, respectively.

Second, long-term collaborative learning among multiple neural receivers is indispensable for each of them to acquire channel knowledge efficiently. To this end, a graph-based collaborative learning scheme called GraphRx is developed to retrain uplink neural receiver collaboratively among BSs. First, considering a collaboration graph among BSs, GraphRx is formulated as a personalized federated learning problem, wherein the graph weights and neural receiver models are learned together so that generalization and personalization are jointly optimized. Second, the problem is solved through an alternating approach under the federated learning paradigm. Particularly, an approximate generalization bound is derived to enable graph optimization at the server without accessing local data on BSs. To reduce the overhead of training pilots, data augmentation is employed. GraphRx is evaluated via extensive simulation. Key parameters of GraphRx are first found through an ablation study. Next, the effectiveness of the approximation in the generation bound is validated. Comparisons with the state-of-the-art schemes are finally conducted. Results show that, given the same coded bit error rate, GraphRx achieves an SNR gain of 0.4–0.9 dB and 0.5–2.1 dB for the cases without and with inter-cell interference, respectively.

In summary, this dissertation identifies four critical research problems at the network infrastructure level, resource management level, and link level, respectively. The AI-assisted approaches are developed correspondingly to address each of these problems, where they function collectively to boost network throughput significantly.

20011372

摘要

为了支持数字孪生、元宇宙等多样化的高带宽应用，下一代无线网络亟需实现极高的吞吐量。同时，人工智能（AI），尤其是深度学习方法，将在未来网络设计中发挥不可或缺的作用。鉴于提升吞吐量和将AI深度融入无线网络的迫切需求，本文致力于在不同架构层面设计基于AI的关键算法，以显著提升整体网络吞吐量。整体设计框架涵盖三个架构层面，即网络基础设施层面、资源管理层面和链路层面。在每一层面，本文均提出了关键研究问题并设计了相应的AI算法加以解决。

在网络基础设施层面，随着基站部署密度不断增加，相较于光纤回传，无线回传方式具备更高的成本效益和灵活性。因此，如何构建高吞吐量的无线回传网成为一个亟需研究的重要问题，对于确保未来网络（尤其是6G高频段网络）的覆盖性和连接性至关重要。为维持高吞吐量，需要通过添加中继节点对回传网结构进行增强。然而，中继节点的部署位置选择面临两个挑战：其一，缺乏刻画网络结构变化与吞吐量增益关系的理论；其二，最优中继部署是一个复杂的组合优化问题。为此，本文首先建立了基于团（clique）的瓶颈理论，通过构建网络的团瓶颈结构来确定吞吐量，并进一步计算clique 梯度以量化各团对整体吞吐量的影响。在此基础上，将clique 梯度嵌入深度强化学习（DRL）框架中，通过动作掩码策略仅选择具有最高clique 梯度的中继位置，从而解决了第二个挑战。大量仿真结果表明，该基于DRL的中继部署方案（DeepRP）可使网络吞吐量提升超过50%，较基线方案提高10.4%至32.1%。

在资源管理层面，网络切片是支撑多样化业务和高效管理物理网络资源的关键技术。实现无线接入网的切片时，最具挑战性的任务之一是在多个小区之间实施切片的空口资源分配。其挑战主要包括：第一，分配给不同切片的资源必须在链路层面实现软隔离；第二，即使通信链路遭遇衰落和干扰，也必须满足用户的差异化服务质量（QoS）需求；第三，必须遵守长期统计意义上的切片策略（包括非均衡策略）。为此，本文首先将链路层切片执行建模为一个资源分配优化问题，目标是在满足软隔离、QoS 保证及长期策略约束的前提下最小化无线资源消耗。随后，本文设计了一种基于DRL的两阶段迭代算法LinkSlice：第一阶段利用DRL结合图神经网络（GNN）确定链路传输速率以捕捉干扰关系；第二阶段在给定传输速率的基础上分配切片资源。实验结果表明，LinkSlice 能够快速收敛至接近最优解，成功解决了链路层切片执行的三大挑战，并进一步将网络吞吐量提升18.5%。

在链路层面，基于神经网络的无线接收机（Neural Rx）是AI原生通信最具潜力的方向之一。上行Neural Rx 通常经过离线预训练后部署于基站，但其性能在动态信道环境下会显著下降，因此需要高效的在线训练机制以维持性能。本文从两个方面研究了Neural Rx 的在线学习：1）如何实现短期快速适应；2）如何提升长期协作学习下的泛化能力。针对这两方面，本文首先设计了面向模块化Neural Rx（即神经网络信道估计器）的快速适应方案，随后提出了多基站端到端Neural Rx 的长期协作学习框架，具体如下。

首先，必须对神经网络信道估计器进行短期快速的在线适应。传统方法通常依赖真实信

20011372

道系数作为监督标签，但此类标签在实际在线环境中不可获得，因此带来了重大挑战。为了解决这一问题，本文在原有的信道估计任务之上引入了一项自监督任务，以实现无标签的自适应训练。具体而言，该任务通过在每个接收帧中随机掩码部分资源单元并重建这些被掩盖部分来完成。为了实现有效的重建，任务输入必须同时包含两部分：未被掩盖的资源单元以及被掩盖单元对应的数据符号估计。这些估计符号通过一个在线符号恢复机制获得，因此不会引入额外的导频开销。为整合主任务与辅助任务，本文设计了一种双分支掩码自编码器（MAE）模型，称为ChannelMAE。其中，每个分支分别对应一个任务，两分支共享同一个特征编码器，但使用不同的解码器。在在线适应过程中，通过优化自监督分支来更新共享编码器，该分支能够学习信道的统计特征，并将其共享给信道估计分支，从而显著提升在线信道估计的准确性。大量实验结果表明，ChannelMAE 在信道估计误差方面显著优于现有方法：与离线预训练模型相比，误差降低高达71.8%；与最新的自适应方法相比，误差降低高达87.1%。

其次，端到端Neural Rx 的长期多点协作学习对于高效获取线上信道知识和提升泛化性能至关重要。为此，本文提出了一种基于合作图的协作学习方案GraphRx，用于多基站间协同训练上行Neural Rx。首先，基于基站之间的合作图，GraphRx 被建模为一个个性化联邦学习问题，在该问题中，合作图权重和接收机模型被联合优化，以在模型的泛化能力与个性化性能之间取得平衡。在联邦学习范式下，该问题通过交替优化方法加以求解。特别地，本文推导并近似了一个模型泛化界，使服务器能够在不访问各基站本地数据的情况下优化合作图权重。与此同时，为降低训练过程中的导频开销，GraphRx 在多点协作训练中引入了数据增强机制。本文通过大量仿真对GraphRx 进行了系统评估：先通过消融实验确定了GraphRx 的关键参数；之后验证了所推导近似泛化界在实际应用中的有效性；最后将GraphRx 与最新方法进行了对比。结果表明，在相同的编码误码率下，GraphRx 在无小区间干扰和有小区间干扰的场景中分别实现了0.4–0.9 dB 和0.5–2.1 dB 的信噪比增益，显著优于现有方法。

综上所述，本文在网络基础设施层面、资源管理层面和链路层面依次提出并解决了四个关键问题，设计了相应的AI 算法框架。这些方法在不同层面协同作用，最终显著提升了网络吞吐量。

20011372

Contents

Abstract **iii**

List of Figures **xiii**

List of Tables **xv**

Nomenclature **1**

1 Introduction **3**

1.1 AI for Wireless Networks: Boosting Throughput at Three Architectural Levels 3
1.2 Research Problems and Challenges 5
1.2.1 Network Infrastructure Level: Relay Placement for Wireless Backhaul Networks 5
1.2.2 Resource Management Level: Slice Enforcement for Multi-Cell Radio Access Networks . . 7
1.2.3 Link Level: Online Learning of Uplink Neural Receivers 9
1.3 Related Work 12
1.3.1 Relay Placement for Multi-Hop Wireless Networks 12
1.3.2 Network Slicing for Radio Access Networks 14
1.3.3 Design of Neural Receivers 16
1.4 Developed Approaches and Organization of the Dissertation 18

2 Relay Placement for Wireless Backhaul Networks Based on Reinforcement Learning **23**

2.1 System Model 24
2.1.1 Network Model 24
2.1.2 Link Conflict Graph and Maximal Clique 25
2.2 Clique-Based Bottleneck Theory 26
2.2.1 Clique-Based Bottleneck Structure 26
2.2.2 Clique Gradient 29
2.3 DRL-based Relay Placement with Clique-based Bottleneck Theory 31
2.3.1 Relay Placement Problem 31
2.3.2 Key Designs of DeepRP 32
2.3.3 DRL Training and Inference 35
2.4 Performance Evaluation 36
2.4.1 Simulation Setup 36

20011372

2.4.2 Validation of DeepRP . . . 38
2.4.3 Comparisons with Baseline Approaches . . . 40
2.5 Summary . . . 44

3 Network Slice Enforcement for Radio Access Networks Based on Reinforcement Learning 45
3.1 System Model . . . 46
3.1.1 5G Multi-Cell RAN . . . 46
3.1.2 Slice Service Model . . . 48
3.1.3 Interference Graph Model . . . 49
3.2 Problem Formulation and Transformation . . . 49
3.2.1 Problem Formulation . . . 49
3.2.2 Problem Transformation . . . 51
3.3 LinkSlice Overview . . . 53
3.4 Two-Stage Slice Enforcement: A DRL-Based Heuristic Solution . . . 54
3.4.1 Stage I: A DRL Approach for Interference-Aware Rate Determination . . . 54
3.4.2 Stage II: Slice Enforcement Through Link-Level Scheduling . . . 59
3.4.3 Overall Algorithm of LinkSlice . . . 62
3.5 Performance Evaluation . . . 62
3.5.1 Simulation Setup . . . 62
3.5.2 Convergence of LinkSlice . . . 65
3.5.3 Performance Compared with Optimal Solutions . . . 66
3.5.4 Key Parameter Determination . . . 66
3.5.5 Performance Compared with Existing Schemes . . . 68
3.6 Summary . . . 72

4 Self-Supervised Learning Assisted Online Adaptation of Neural Channel Estimators 73
4.1 System Model . . . 74
4.2 Overview of Two-Task Learning Framework . . . 74
4.2.1 Task Formulation . . . 74
4.2.2 Overview of Two Learning Phases . . . 75
4.3 Key Designs of ChannelMAE . . . 76
4.3.1 Input Pre-Processing . . . 76
4.3.2 Model Architecture: Two-Branch MAE . . . 77
4.3.3 Two-Phase Training Procedures . . . 79
4.4 Performance Evaluation . . . 82
4.4.1 Simulation Setup . . . 82
4.4.2 Validation of Online Performance Degradation and Online Model Convergence . . . 84
4.4.3 Ablation Studies . . . 85
4.4.4 Comparison with Baselines . . . 89
4.5 Summary . . . 91

20011372

5 Online Collaborative Learning among Multiple Cells for Uplink Neural Receivers 93
- 5.1 System Model . . . 94
  - 5.1.1 SIMO-OFDM Communications . . . 94
  - 5.1.2 Design of Neural Receiver . . . 94
  - 5.1.3 Elements of Supervised Learning Problem . . . 95
- 5.2 Problem Formulation and Analysis . . . 96
  - 5.2.1 Collaboration-Graph-Based Learning . . . 96
  - 5.2.2 Problem Analysis . . . 97
- 5.3 Collaboration Graph Optimization . . . 98
  - 5.3.1 Derivation of Generalization Bound . . . 98
  - 5.3.2 Empirical Objective for Optimizing Collaboration Graph . . . 100
- 5.4 Graph-Based Collaborative Learning . . . 101
  - 5.4.1 Alternating Optimization . . . 101
  - 5.4.2 Algorithm Design . . . 102
  - 5.4.3 Mechanism of Data Augmentation . . . 103
- 5.5 Performance Evaluation . . . 103
  - 5.5.1 Simulation Setup . . . 103
  - 5.5.2 Ablation Study . . . 105
  - 5.5.3 Effectiveness of Divergence Approximation . . . 108
  - 5.5.4 Comparison with Baselines . . . 110
- 5.6 Summary . . . 112

6 Conclusions and Future Work 113
- 6.1 Conclusions . . . 113
- 6.2 Future Work . . . 115

Bibliography 127

A Proof of Theorems in Chapter 2 129

B Proof of Theorems in Chapter 3 135

C Proof of Theorem in Chapter 5 137

Publication List 139

Acknowledgments 141

20001372

List of Figures

1.1 Backhaul mesh network with network bottlenecks. 5
1.2 RAN slicing framework in 5G scenarios. 8
1.3 Slice-level enforcement with unbalanced slicing policies in multi-cell RAN slicing. 9
1.4 Fully-learned and partially-learned neural receivers. 10
1.5 Multi-cell network with heterogeneous channel scenarios. 12
1.6 Organization of this dissertation. 18

2.1 Link conflict graphs and maximal cliques. 25
2.2 Clique-based bottleneck structure. 27
2.3 DeepRP: DRL Design Framework 32
2.4 Episode rewards of DRL training. 38
2.5 Network throughput achieved by DeepRP. 39
2.6 Network throughput with different flow rerouting methods. 39
2.7 Impact of various path loss exponent values. 40
2.8 Comparisons with the exhaustive search in small-scale problems. 41
2.9 Network throughput with various numbers of deployed relays. 41
2.10 Network throughput with various numbers of flows. 41
2.11 Network throughput gain over the original network under various candidate relay site densities. . . 42
2.12 Depth of clique-based bottleneck structures. 43
2.13 Comparison of clique-based bottleneck structures. 43

3.1 BS deployment and interference graph in multi-cell RAN. 48
3.2 LinkSlice overview: a DRL-based heuristic solution for slice enforcement. 54
3.3 LinkSlice detailed design: two-stage slice enforcement. 57

3.4 Convergence performance of LinkSlice in the first 120 episodes with a different number of BSs. . 63

3.5 Comparisons with the optimal solutions to P1 and P2. 64

3.6 Slicing policy conformance and network performance with various penalty coefficients. 65

3.7 Tradeoff between network throughput and sample efficiency with various sector sizes. 67

3.8 Network throughput in the uniform user distribution. 68

3.9 Network throughput in the cell-center-biased user distribution. 68

3.10 Network throughput in the cell-edge-biased user distribution. 69

3.11 QoS satisfaction in the saturation scenario of LinkSlice. 69

3.12 CDF plot of link data rate distribution for Slice C (1Mbps) in the saturation scenario of LinkSlice. 70

3.13 Comparisons of slicing policy conformance. 70

3.14 Number of interfering PRBs among different slices with various unbalance indices. 71

3.15 Saturation throughput under slicing policies with various unbalance indices. 71

4.1 Overview of two learning phases. 75

4.2 Detailed designs of two model branches. 78

4.3 Model architecture of ChannelMAE. 80

4.4 Uncoded and channel-coded online symbol-recovery schemes. 81

4.5 Performance degradation of the pretrained model encountering the offline-online distribution shift. 84

4.6 Online losses of ChannelMAE versus online batch index (both losses are running-averaged for 30 batches). 84

4.7 Various model architectures of ChannelMAE. 85

4.8 Ablation studies of input pre-processing. 85

4.9 Comparison of two online symbol-recovery schemes in terms of channel estimation performance and computation cost. 86

4.10 Comparison of various data augmentation times and online-adaptation SNR values. 87

4.11 Comparison of online model updating strategies. 87

4.12 Evaluation of offline-pretrained models and conventional baselines in offline channel environments. 88

4.13 Evaluation of online-adapted models and conventional baselines in online channel environments. . 89

4.14 Continual online adaptation in multiple channel environment changes. 90

4.15 Re-evaluation of online-adapted ChannelMAE in offline channels. 90

5.1 Fully convolutional network architecture of neural receiver. 94

20001372

5.2 Working process of each BS in collaborative learning. 95

5.3 GraphRx: graph-based collaborative learning scheme. 97

5.4 Average validation BER at various communication rounds. 105

5.5 Average validation BER with various distance metrics and complexity indicator values (i.e., C). . 106

5.6 Average validation BER with directed/undirected graphs. 106

5.7 Visualization of directed collaboration graphs. 107

5.8 Coded BER of GraphRx tested under various augmentation times. 108

5.9 Correlations between model distance and estimated divergence. 109

5.10 Coded BER tested without inter-cell interference. 110

5.11 Coded BER tested with inter-cell interference. 110

5.12 Standard deviation of coded BER tested for Hetero-2. 111

20011372

20011372

List of Tables

3.1 Key simulation parameters. 65

4.1 Comparison of model parameter counts (in millions). 88

5.1 Heterogeneous multi-cell environments 104

20011372

Nomenclature

$\bar{\boldsymbol{\phi}}_m^t$	Aggregated model at BS m in round t
$\beta_{u,i}^{(b,m)}$	Nominal SNR for user u on PRB i within slice m and BS b
$\boldsymbol{\beta}_u(t)$	Vector of nominal SNR for user u at time step t
$\boldsymbol{\theta}_{\mathrm{A}}$	Parameters of the actor network
$\boldsymbol{\theta}_{\mathrm{V}}$	Parameters of the critic network
$\boldsymbol{\alpha}_{\mathrm{e}}$	Shared encoder
$\boldsymbol{\beta}_{\mathrm{a}}$	Auxiliary-task decoder
$\boldsymbol{\beta}_{\mathrm{m}}$	Main-task decoder
$\boldsymbol{\theta}_m$	Personalized neural receiver model at BS m
$\boldsymbol{\theta}_m^{t,i}$	Model of BS m after i-th mini-batch gradient descent in round t
$\boldsymbol{\theta}_{\mathrm{a}}$	Auxiliary-task model
$\boldsymbol{\theta}_{\mathrm{m}}$	Main-task model
Δr_f	Perturbation on flow f's rate
ΔR_j	Perturbation on clique j's equivalent capacity
Δs_k	Perturbation on clique k's fair share
$\Delta_{\mathrm{R},u}$	Gap between QoS requirement and actual data rate for user u
$\delta_u^{(b,m)}$	SNR degradation for cell-edge user u

20011372

$\ell(\cdot)$	Binary cross-entropy (BCE) loss
$\ell_{\mathrm{a}}(\cdot)$	Auxiliary-task loss function
$\ell_{\mathrm{m}}(\cdot)$	Main-task loss function
η	Learning rate
γ	Discount factor for accumulated rewards
$\gamma_{u,i}^{(b,m)}$	SINR of user u on PRB i within slice m and BS b
$\hat{\mathbf{G}}_{\mathrm{p}}$	Expanded least-square estimate matrix
$\hat{\mathbf{H}}_{\mathrm{p}}$	Least-square estimate of channel coefficients at pilot positions
$\hat{\mathbf{X}}$	Detected transmit symbols
$\hat{B}(\cdot)$	Empirical optimization objective for collaboration graph
$\hat{d}(\cdot)$	Approximated pair-wise divergence between models
$\hat{F}(\cdot)$	Empirical risk function over the local dataset
λ	Penalty coefficient for slicing policy violation
$\mathbb{E}[\cdot]$	Expectation operator
$\mathbf{a}(t)$	Action vector at time step t
$\mathbf{D}$	Distribution divergence matrix
$\mathbf{F}$	Linear interpolation matrix for LMMSE estimation
$\mathbf{G}$	Inter-cell interference matrix
$\mathbf{g}(j)$	Clique gradient for clique j
$\mathbf{g}_{kl}$	Inter-cell interference on subcarrier k and symbol l
$\mathbf{H}$	Channel coefficients
$\mathbf{h}_{kl}$	Channel coefficients on subcarrier k and symbol l
$\mathbf{I}$	Identity matrix

20011372

$\mathbf{K}$	Keys in self-attention network
$\mathbf{L}$	Preknown training pilot bits
$\mathbf{M}_\mathrm{p}$	Pilot mask
$\mathbf{M}_\mathrm{r}$	Random binary mask for auxiliary task
$\mathbf{N}$	Additive white Gaussian noise
$\mathbf{N}_\mathrm{a}$	Additive noise for data augmentation
$\mathbf{n}_{kl}$	Noise on subcarrier k and symbol l
$\mathbf{p}$	Data quantity vector
$\mathbf{Q}$	Queries in self-attention network
$\mathbf{R}_{\mathbf{H},\mathbf{H}_\mathrm{p}}$	Cross-correlation matrix between ground-truth channel and pilot channel
$\mathbf{R}_\mathrm{o}$	Observed received signal divided by detected transmit symbols
$\mathbf{R}_{\mathbf{H}_\mathrm{p},\mathbf{H}_\mathrm{p}}$	Auto-correlation matrix of pilot channel
$\mathbf{s}_t$	State vector at step t
$\mathbf{s}_u(t)$	State of user u at time step t
$\mathbf{S}_\mathrm{a}$	Sequence of tokens output by the shared encoder for auxiliary task
$\mathbf{S}_\mathrm{m}$	Sequence of tokens output by the shared encoder for main task
$\mathbf{T}$	Stacked input of neural receiver
$\mathbf{T}_m^{(i)}$	Input of the i-th data instance at BS m
$\mathbf{T}_\mathrm{a}$	Tokenized input for auxiliary task
$\mathbf{T}_\mathrm{m}$	Tokenized input for main task
$\mathbf{V}$	Values in self-attention network
$\mathbf{W}$	Directed graph weight matrix of collaboration graph
$\mathbf{W}_i^K$	Learnable parameters for keys in attention head i

$\mathbf{W}_i^Q$	Learnable parameters for queries in attention head i
$\mathbf{W}^t$	Collaboration graph weights in round t
$\mathbf{W}_i^V$	Learnable parameters for values in attention head i
$\mathbf{W}^O$	Output projection matrix in self-attention network
$\mathbf{w}_m$	Collaboration vector of BS m
$\mathbf{X}$	Transmit signal matrix
$\mathbf{X}_\mathrm{p}$	Pilot symbols
$\mathbf{X}_\mathrm{R}$	Transmitted demodulation reference symbols (DMRS) configuration matrix
$\mathbf{Y}$	Received signals
$\mathbf{Y}_\mathrm{a}$	Augmented received signal frame
$\mathbf{Y}_\mathrm{p}$	Received pilot symbols
$\mathbf{Y}_\mathrm{o}$	Observed received signal with random mask
$\mathbf{y}_{kl}$	Frequency-domain received signal on subcarrier k and symbol l
$\hat{\mathcal{P}}_f$	Predecessor vertices of flow f
$\hat{\mathcal{R}}$	Set of relay nodes
$\hat{\mathcal{R}}_{can}$	Set of candidate relay sites
$\mathcal{B}$	Set of base stations
$\mathcal{B}_k$	Successor vertices of clique k
$\mathcal{D}$	Set of routes
$\mathcal{E}_b$	Edge set of the bottleneck structure graph
$\mathcal{E}_c$	Edge set of the link conflict graph
$\mathcal{E}_w$	Edge set of the full virtual network topology
$\mathcal{E}_\mathrm{I}$	Set of edges in the interference graph

$\mathcal{F}$	Set of flows
$\mathcal{F}_l$	Set of flows traversing link l
$\mathcal{G}$	Directed collaboration graph
$\mathcal{G}_b$	Bottleneck structure graph
$\mathcal{G}_c$	Link conflict graph
$\mathcal{G}_w$	Full virtual network topology
$\mathcal{G}_{\mathrm{I}}$	Interference graph
$\mathcal{H}$	Hypothesis space
$\mathcal{K}$	Set of maximal cliques
$\mathcal{L}$	Set of directed links
$\mathcal{L}_k$	Set of links in clique k
$\mathcal{L}_{\mathrm{off}}(\cdot)$	Offline loss function
$\mathcal{L}_{\mathrm{on}}(\cdot)$	Online loss function
$\mathcal{M}$	Set of admitted network slices
$\mathcal{N}$	Set of network nodes
$\mathcal{N}_m$	Neighbor set of BS m
$\mathcal{P}_k$	Predecessor vertices of clique k
$\mathcal{R}_f$	Route of flow f
$\mathcal{U}$	Set of users in the RAN
$\mathcal{U}_{\mathrm{c}}$	Set of all cell-center users
$\mathcal{U}_{\mathrm{e}}$	Set of all cell-edge users
$\mathcal{U}_b$	Set of users associated with BS b
$\mathcal{U}_{b,\mathrm{c}}$	Set of cell-center users associated with BS b

$\mathcal{U}_{b,\mathrm{e}}$	Set of cell-edge users associated with BS b
$\mathcal{U}_{b,m}$	Set of users associated with BS b and slice m
$\mathcal{V}$	Set of local models in the collaboration graph
$\mathcal{V}(u)$	Set of neighboring links of user u in the interference graph
$\mathcal{V}_b$	Vertex set of the bottleneck structure graph
$\mathcal{V}_c$	Vertex set of the link conflict graph
$\mathcal{V}_w$	Node set of the full virtual network topology
$\mathcal{V}_\mathrm{I}$	Set of nodes in the interference graph
$\mathcal{W}_m$	Set of users associated with slice m
$\mathcal{Z}_m$	Local dataset of BS m
$\mathrm{Att}(\cdot)$	Multi-head self-attention layer
$\mathrm{cov}(\cdot)$	Covariance operator
$\mathrm{GeLu}(\cdot)$	Gaussian Error Linear Unit activation function
$\mathrm{Softmax}(\cdot)$	SoftMax activation function
$\mathrm{Tr}(\cdot)$	Trace of a matrix
$\mathrm{vec}(\cdot)$	Vectorization of a matrix
μ	Penalty for infeasibility of the second stage
ϕ	Rotation angle for data augmentation
σ^2	Additive Gaussian noise variance
σ_a^2	Variance of additive noise for data augmentation
σ_n	Standard deviation of Gaussian noise
σ_X	Standard deviation of shadowing effect
σ_x	Transmit power of each symbol

σ_{ds}	Root-mean-squared (RMS) delay spread
$\sigma_{\mathrm{R}}(\cdot)$	ReLU activation function
$\sigma_{\mathrm{S}}(\cdot)$	SoftMax activation function
$\tilde{\gamma}_{u,i}^{(b,m)}$	SINR lower bound for user u on PRB i within slice m and BS b
$\tilde{\mathbf{s}}_u(t)$	Embedded state of user u at time step t
$\tilde{r}(t)$	Instantaneous reward at time step t
$\tilde{R}_u$	Link data rate for user u
A	Augmentation factor
$A^{\pi}(t)$	Advantage function at time step t
A_t^{π}	Advantage function at step t
$a_k(t)$	Action for sector k at time step t
a_t	Action at step t
c	Speed of light
C_l	Capacity of link l
c_l	Link cost for link l
d	Propagation distance
d_0	Reference distance
d_k	Key dimension in self-attention network
$d_{\mathcal{H}\Delta\mathcal{H}}(\cdot)$	$\mathcal{H}\Delta\mathcal{H}$-divergence between distributions
F	Number of flows
$f(\cdot)$	CQI to MCS mapping function
$F^{-1}(\cdot)$	Reverse mapping function from rate to SINR
f_c	Carrier frequency

20011372

$F_m(h)$	Expected risk of hypothesis h over true data distribution $\mathcal{D}_m$
G_r	Receive effective gain
G_t	Transmit effective gain
H	Number of highest-gradient cliques
h	Number of attention heads
$h(t)$	Channel impulse response
h_m^*	Optimal minimizer of expected risk at BS m
$h_{u,i}^{(b,m)}$	Channel coefficient between user u and BS b on PRB i within slice m and BS b
$I_{u,i}^{(b,m)}$	Maximum tolerance interference for user u on PRB i within slice m and BS b
K	Number of cliques
K_s	Number of virtual sectors
$k_{abs}(\cdot)$	Frequency-dependent absorption coefficient
L	Number of links
$l(\cdot)$	SINR to CQI mapping function
M	Number of base stations
N	Number of deployed nodes
N_b	Number of bits per symbol
N_f	Number of subcarriers in one frame
N_RS	Number of DMRS symbols within one TTI
N_r	Number of receive antennas at the BS
N_s	Number of sectors per cell
N_sc	Number of PRBs within one slot
N_m	Number of data instances at BS m

N_r	Number of relays to be deployed
N_{aux}	Number of unmasked OFDM symbols in auxiliary task
N_{da}	Number of ResNet blocks in auxiliary-task decoder
N_{dm}	Number of ResNet blocks in main-task decoder
N_{e}	Number of encoder blocks
N_{fp}	Number of subcarriers carrying pilots
N_{hid}	Hidden layer dimension in MLP block
N_{sp}	Number of OFDM symbols carrying pilots
N_{s}	Number of OFDM symbols in one frame
N_{PRB}	Total number of PRBs in a radio frame
P_{c}	Transmit power for cell-center users
P_{e}	Transmit power for cell-edge users
P_u	Transmit power for user u
P_{N}	Noise power on each PRB
$p_{b,m}$	Long-term slicing policy for slice m on BS b
$p_{k,l}$	Virtual scheduling variable for link l in clique k
r_f	Flow rate of flow f
R_k	Clique equivalent capacity for clique k
R_m	Minimum data rate requirement for slice m
r_t	Reward at step t
$r_{u,i}^{(b,m)}$	Transmission rate for user u on PRB i within slice m and BS b
s_k	Clique fair share for clique k
T	Network throughput

20001372

T_{f}	Number of time slots in a radio frame
T_{c}	Number of communication rounds
T_{s}	Slicing policy period in radio frames
v	Terminal velocity
$V^{\pi}(\cdot)$	State value function
w_{mj}	Collaboration weight from BS j to BS m
$w_{u,v}$	Weight of edge between nodes u and v in the interference graph
X	Log-normal random variable for shadowing effect
x_{kl}	Transmit symbol on subcarrier k and symbol l
$x_{u,i}^{(b,m)}$	PRB allocation indicator for user u on PRB i within slice m and BS b
3GPP	3rd Generation Partnership Project
6G	Sixth generation
BCE	Binary cross-entropy
BER	Bit error rate
BLER	Block error rate
BS	Base station
CN	Core network
CNN	Convolutional neural network
CQI	Channel quality indicator
CU	Central unit
DAG	Directed acyclic graph
DL	Deep learning
DMRS	Demodulation reference signal

DNN	Deep neural networks
DRL	Deep reinforcement learning
DU	Distributed unit
FCNN	Fully connected neural network
FDD	Frequency division duplexing
FL	Federated learning
GAE	Generalized advantage estimation
GCN	Graph convolutional network
GeLU	Gaussian error linear unit
GNN	Graph neural network
IQCP	Integer quadratic-constrained programming
IRS	Intelligent reflective surface
LDPC	Low-density parity-check
LLR	Log-likelihood ratio
LMMSE	Linear minimum-mean-square-error
LN	Layer normalization
LS	Least-square
MAC	Medium access control
MAE	Masked autoencoder
MCS	Modulation and coding scheme
ML	Machine learning
MLP	Multi-layer perceptron
MMSE	Minimum mean-squared error

20011372

MSE	Mean-squared error
OFDM	Orthogonal frequency division multiplexing
OFDMA	Orthogonal frequency division multiple access
PFL	Personalized federated learning
PHY	Physical layer
PNO	Physical network operator
PPO	Proximal policy optimization
PRB	Physical resource block
QAM	Quadrature amplitude modulation
QoS	Quality of services
RAN	Radio access network
RE	Resource element
ReLU	Rectified linear unit
RMS	Root mean squared
S-V	Saleh-Valenzuela
SER	Symbol error rate
SIMO	Single-input-multiple-output
SINR	Signal-to-interference-and-noise ratio
SISO	Single-input-single-output
SLA	Service level agreement
SNR	Signal-to-noise ratio
SSL	Self-supervised learning
TDD	Time division duplexing

20001372

TDL	Tapped delay line
THz	Terahertz
TTI	Transmission time interval
TTT	Test-time training
UE	User equipment
UMa	Urban macro
UMi	Urban micro
VC	Vapnik-Chervonenkis
VNO	Virtual network operator

20001372

20011372

Chapter 1

Introduction

1.1 AI for Wireless Networks: Boosting Throughput at Three Architectural Levels

The sixth generation (6G) wireless networks are envisioned to be deployed and commercialized in the 2030s (6G Flagship, 2025; Letaief et al., 2019). A wide range of research initiatives are proposed worldwide in academia and industry to explore promising aspects of 6G (6G Flagship, 2025; Letaief et al., 2019; Wang et al., 2023). It is expected that the next-generation wireless networks can achieve extremely high data rates exceeding 100 Gigabits per second (Gbps), low latency of less than 1 millisecond (ms), and ultra-enhanced connectivity. In such future networks, a growing number of diverse applications has been proposed, such as Metaverse (Wang et al., 2023), augmented reality/virtual reality (Giordani et al., 2020; Wang et al., 2023), and holographic telepresence (Giordani et al., 2020). The above potential applications are bandwidth-intensive or throughput-hungry: they require the network to sustain a high network throughput. For instance, vast amounts of data such as complex 3D graphics must be transmitted to create a virtual environment in Metaverse. For truly immersive remote interaction that stimulates all five human senses, these sensory data will further raise the total throughput requirements of future networks. Therefore, to support these promising applications in future wireless networks, boosting throughput is one of the most critical aspects, and it is the focus of this dissertation.

Artificial Intelligence (AI)-based methods will play a crucial role in future wireless networks (Zhang et al., 2019; Liu et al., 2023; Giordani et al., 2020). 6G networks are envisioned to be AI-native (Letaief et al., 2019), where AI will not only be an alternative tool but an indispensable component of wireless networks. This is because future wireless networks will reach a level of complexity that renders the traditional model-based methodology insufficient (Zhang et al., 2019; Liu et al., 2023), and this motivates us to equip future wireless networks with AI capabilities.

Therefore, this dissertation aims to develop AI-assisted algorithms for boosting network throughput instead of conventional methods. However, this is a comprehensive research problem that must be carefully decomposed into sub-problems, since network throughput is a complicated metric influenced by diverse designs at various architectural levels. To fulfill such decomposition, we adopt a coarse-to-fine design paradigm, focusing sequentially on three critical architectural levels: network infrastructure level, resource management level, and link level at the physical layer (PHY). They are elaborated in detail as follows.

- **Network infrastructure level:** This foundational level provides the physical and logical structures necessary for network operation, which can significantly impact the overall capacity. At this level, we especially consider a backhaul network infrastructure with multi-hop wireless connectivity, which enjoys lower deployment costs, higher flexibility, and better signal coverage than the wired counterpart. The problem of how to augment a multi-hop backhaul network by placing backhaul relays into the network is studied. By exploiting AI algorithms, an intelligent relay placement scheme can be obtained, which can significantly boost the overall network throughput.

- **Resource management level:** A high-capacity network infrastructure provides the foundation for deploying diverse services and optimizing their resources. This level is focused on how to optimize resource management for diverse network services. At this level, we especially consider the design of radio access network (RAN) slicing. By applying AI algorithms here, network resources can be dynamically allocated in a way that maximizes resource utilization efficiency while satisfying service requirements, equivalent to boosting network throughput with a given number of resources.

- **Link level:** Effective service-level management ensures that network traffic is efficiently allocated onto the physical layer at the link level. This level is focused on designing advanced physical layer technologies to improve link-level throughput by reducing bit errors. At this level, we especially investigate neural-network-based wireless receivers. With an optimized infrastructure and effective service management already in place, network throughput can be further boosted as each link of this network achieves a higher link-level throughput.

In summary, leveraging AI-assisted algorithms, these three levels, i.e., network infrastructure level, resource management level, and link level, are significantly advanced and optimized, operating in concert to boost overall network throughput.

20011372

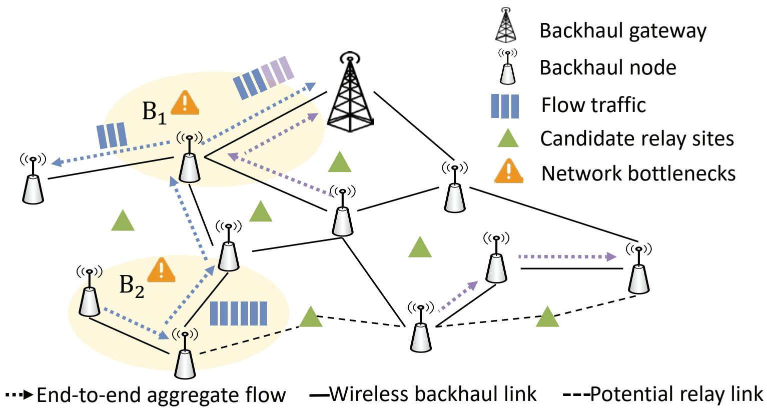


Figure 1.1: Backhaul mesh network with network bottlenecks.

1.2 Research Problems and Challenges

As stated above, the objective of this dissertation is to boost network throughput at three different architectural levels via AI-assisted methods. The research problems and their key challenges are presented for each level respectively as follows.

1.2.1 Network Infrastructure Level: Relay Placement for Wireless Backhaul Networks

Research Problem

We especially consider a network infrastructure with multi-hop wireless connectivity, which enjoys lower deployment costs, higher flexibility, and better signal coverage than the wired counterpart. However, it is challenging to design the infrastructure of a multi-hop wireless backhaul network (i.e., backhaul mesh) to achieve high network throughput. This is because network bottlenecks arise in various congestion regions of the network (Ros-Giralt et al., 2019) that interact with each other. As shown in Fig. 1.1, two regions are overloaded with flow traffic such that they constitute the bottlenecks B_1 and B_2 for these flows. In this case, if one tries to resolve the bottleneck B_1, it is short-sighted to only focus on this single bottleneck without considering the interaction between two bottlenecks. If the formation of B_2 leads to B_1, there will be much more performance gain by resolving B_2 instead of B_1. Although some recent work (Ros-Giralt et al., 2019) studies this interaction among bottlenecks in wired networks, there is no existing work studying the complicated relationship among bottlenecks in wireless networks.

Therefore, at the network infrastructure level, we study the problem of how to design a backhaul mesh network by considering interactions among bottlenecks. Instead of building an entire network from scratch, a network infrastructure augmentation problem is formulated where backhaul relay nodes are dynamically placed in the currently deployed backhaul mesh. The process of network augmentation refers to *relay placement* in the rest of this dissertation, whose design principle perfectly aligns with the developing trends of 6G networking, as there exist various candidate relays that support flexible networking, such as mobile BS, vehicle mounted relays, smart repeaters, and even intelligent reflective surfaces. The candidate location sites of relays are assumed to be known in this problem (Yan et al., 2021; Li et al., 2023), and a set of relays are deployed at the selected candidate sites. Relay placement is considered an online dynamic process where the number of relays and their sites can be adjusted periodically or on demand during network operation. In this way, network throughput can be significantly boosted.

Challenges

To augment the network architecture for higher network throughput by adding relays, challenges lie in three aspects. First, the complicated interactions among network bottlenecks in 6G backhaul mesh must be captured, from which the relationship between the network architecture and the network throughput is determined. In wired networks, such interactions are captured via constructing a directed acyclic graph (DAG) named the bottleneck structure and the quantitative theory of bottleneck structures (QTBS) is developed in (Ros-Giralt et al., 2022, 2021) to quantify the interactions among bottleneck links, and it is named the *link-based bottleneck theory* in the rest of the dissertation. However, this theory cannot be applied to the wireless case as it cannot capture various scenarios of link resource conflicts, such as inter-link interference, time sharing due to beam switching, and time division duplex (TDD) constraints (Wang and Wang, 2023). How to develop the bottleneck theory for wireless networks remains an open problem.

Second, the impact of each bottleneck on the network throughput needs to be accurately quantified. As the objective of relay placement is to boost network throughput with fewer relays, the bottlenecks with the highest impact on the overall network throughput must be identified, which enables the placement of relays to be specifically aimed at resolving these critical bottlenecks.

Third, after identifying the network bottlenecks with a significant impact on network throughput, the relays need to be placed at the optimal sites that are selected from the candidate sites to resolve these bottlenecks. Since network throughput is derived based on bottleneck theory in this dissertation, the objective of throughput maximization cannot be expressed analytically as a differentiable function. Traditional optimization techniques cannot address this problem, and thus a learning-based relay placement scheme is called for.

20011372

1.2.2 Resource Management Level: Slice Enforcement for Multi-Cell Radio Access Networks

Research Problem

At the resource management level, to meet diverse service demands of future networks, network slicing techniques (Afolabi et al., 2018; Foukas et al., 2017; Zhang et al., 2017) have been proposed as a key enabler to provide efficient network management. Network operators split a shared physical infrastructure into several virtual networks, each of which consists of multiple network slices. As a building block of an end-to-end virtual network, a network slice spans both core networks (CN) and radio access networks (RAN). Therefore, network slicing falls into two main categories, CN slicing and RAN slicing (3GPP TS 28.542, 2018).

While CN slicing has already been specified by the 3GPP standards (3GPP TS 28.542, 2018), RAN slicing is still in its early stages. In recent years, RAN slicing has attracted great attention from both academic and industrial researchers (3GPP TR 38.801, 2017; 3GPP TS 28.541, 2024; D'Oro et al., 2020; Ferrus et al., 2018). As shown in Fig. 1.2, virtual network operators (VNO) who own and manage network slices need to form payment contracts with physical network operators (PNO) based on the amount of resources allocated to network slices. Such payment contracts are termed service level agreement (SLA) as specified in (3GPP TS 28.542, 2018). Since the users who subscribe to a slice can be distributed across multiple cells, RAN slicing thus involves multiple base stations(Sun et al., 2021; Tang et al., 2019; D'Oro et al., 2019).

RAN slicing mainly consists of two phases in sequence (D'Oro et al., 2019): slicing policy determination and slice enforcement. In the phase of slicing policy determination, both VNOs and PNOs are involved in determining slicing policies for admitted slices. A slicing policy is a type of SLA that prescribes the percentage of resources reserved on base stations for each admitted slice (D'Oro et al., 2019). In the phase of slice enforcement, given the predetermined slicing policy, physical resource blocks (PRBs) on the involved base stations are allocated to each slice. The problem of slicing policy determination is well addressed by many schemes in the literature (Hua et al., 2020; D'Oro et al., 2018; Sexton et al., 2020). Slice enforcement, however, remains under-explored. Therefore, at the resource management level, we study the problem of how to perform slice enforcement in a multi-cell RAN to boost network throughput.

Challenges

Slice enforcement needs to consider several requirements. First, slice isolation needs to be ensured so that the behavior of one slice does not cause performance degradation in another slice (3GPP TR 38.801, 2017). Thus, interference among different slices, i.e., inter-slice interference, needs to be avoided by resource separation

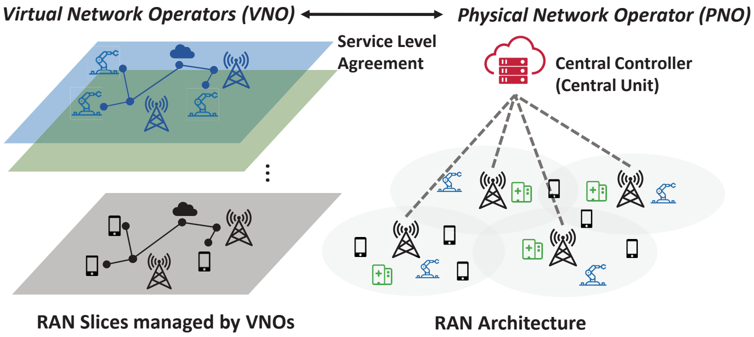

Virtual Network Operators (VNO)
Physical Network Operator (PNO)
Service Level Agreement
Central Controller (Central Unit)
RAN Slices managed by VNOs
RAN Architecture

Figure 1.2: RAN slicing framework in 5G scenarios.

between slices. Second, QoS of users in each slice must be satisfied even under the influence of channel fading and interference. Third, the allocation of PRBs needs to conform to a predetermined slicing policy. This requirement is nontrivial, as slicing policies are periodically determined on a much larger timescale than a radio frame, thus requiring long-term conformance.

To satisfy these requirements, slice enforcement needs to be conducted at the link level rather than at the level of a whole slice. The reason is two-fold. First, slice-level enforcement is focused on the aggregated performance of the slice. It cannot guarantee fine-grained QoS performance of each user, while link-level enforcement can satisfy each user's demands via proper handling of fading and interference on the communication link of the user. Thus, it is desired to allocate resources directly to a link. Second, the number of users served by a slice varies from one cell to another, resulting in unbalanced slicing policies in different cells, as shown in Fig. 1.3. If slice enforcement is carried out per slice (without link-level scheduling) under unbalanced slicing policies, a great amount of resources will be wasted to ensure slice isolation. For example, in Fig. 1.3, if PRBs are allocated to slice A conforming to its slicing policy in cell 1, then most of these PRB resources (i.e., interfering resources in Fig. 1.3) cannot be reused in cells 2 and 3 to avoid inter-slice interference. To solve this problem, slice enforcement at the link level is indispensable.

Link-level slice enforcement makes the requirements of slice enforcement even more challenging, particularly in two aspects. First, slice isolation needs to be conducted at the link level. So far there exist two major types of resource separation: hard isolation and soft isolation. Hard isolation allocates dedicated PRBs to each slice in a network-wide manner but at a great sacrifice of spectral efficiency (Tang et al., 2019). Soft isolation (D'Oro et al., 2019), however, only requires dedicated resources for a slice on interfering base stations. In other words, different slices can reuse the same PRBs unless they are deployed on interfering base stations. Apparently, soft isolation is preferred because of its higher spectrum efficiency. However, how to ensure soft isolation at the link level remains a challenging issue. Second, long-term conformance of slicing policies is closely related

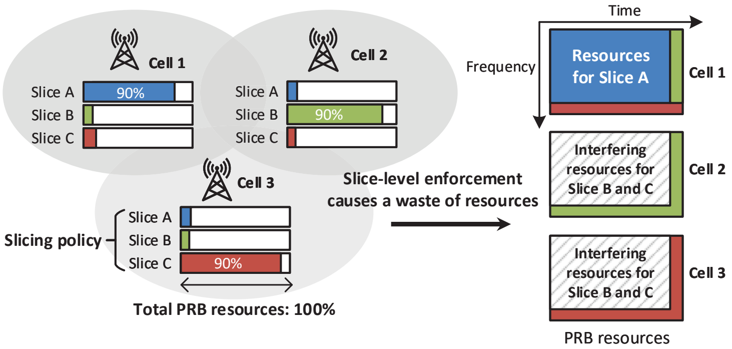


Figure 1.3: Slice-level enforcement with unbalanced slicing policies in multi-cell RAN slicing.

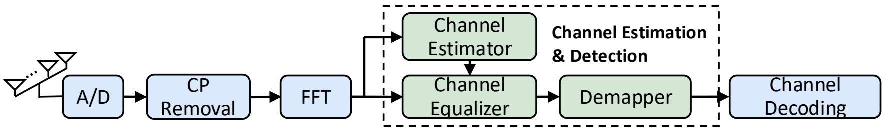
to short-term link-level behavior of slices in each radio frame. Thus, while evaluating long-term conformance of a network slice, link-level performance in each radio frame needs to be characterized, which is still an open problem.

1.2.3 Link Level: Online Learning of Uplink Neural Receivers

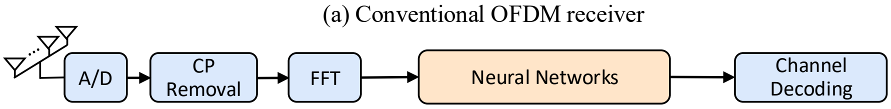
Machine learning (ML) and deep learning (DL)-based wireless PHY design serve as the cornerstone of 6G networks (Honkala et al., 2021). Particularly, the neural network-based wireless receiver, named *neural receiver*, is one of the most promising solutions for 6G receiver design (Honkala et al., 2021). In this dissertation, the term "*neural receiver*" refers to a receiver that uses neural networks (NN) to replace specific function modules or to completely substitute the entire receiver.

As shown in Fig. 1.4(a) and (b), for a fully-learned OFDM neural receiver, all the traditional OFDM receiver modules are replaced by an end-to-end NN. As shown in Fig. 1.4, in a fully-learned neural receiver, the channel estimator and detector are replaced by a neural network that takes the frequency-domain receive signals as inputs and outputs the soft-detection bits. By contrast, for a neural receiver with a partially-learned design, only specific modules are replaced by NN while the remaining ones are still traditionally designed. As shown in Fig. 1.4(c), an NN-based channel estimator is named *neural channel estimator*. It only replaces the conventional channel estimator by an NN, which takes pilot-based channel estimates as inputs and predicts full-frame channel coefficients. This is also termed as a partially-learned neural receiver. A neural receiver, either fully-learned or partially-learned, can significantly outperform traditional schemes without requiring explicit channel modeling or knowledge of channel statistics in real-world scenarios (Honkala et al., 2021; Belgiovine et al., 2021). It can be deployed at a base station (BS) for uplink communications or at a user terminal for downlink communications. This dissertation is focused on OFDM uplink communications, but the designs can also be extended to downlink

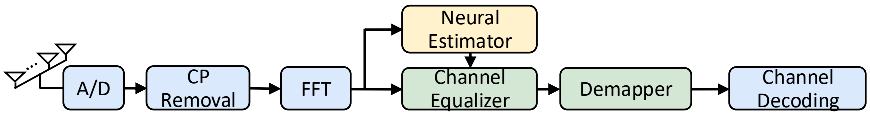
20011372



(a) Conventional OFDM receiver



(b) Fully-learned neural Rx



(c) Partially-learned neural Rx: neural channel estimator

Figure 1.4: Fully-learned and partially-learned neural receivers.

scenarios.

A neural receiver is pre-trained offline and then deployed online at a BS. It is usually impractical for an offline pretraining dataset to cover all possible channel conditions (Fischer et al., 2022; Jiang et al., 2021), so the performance of the pre-trained neural receiver can degrade in varying channel environments. Therefore, it is indispensable for the neural receiver to conduct online adaptation to improve its performance. In other words, neural receivers need to be retrained online.

There are two crucial design aspects for the online adaptation of neural receivers. The first involves conducting efficient, short-term online adaptations to promptly respond to changes in the channel environment. The second focuses on enhancing the long-term online generalization and robustness of neural receiver models. In the following, two problems respectively focused on these two aspects, and their challenges are stated.

Research Problem and Challenges of Short-Term Adaptation

Regarding short-term online adaptation, we focus on studying partially-learned neural receivers, specifically neural channel estimators as shown in Fig. 1.4. This is because the model size of a fully-learned neural receiver is generally much larger than a partially-learned one, and thus retraining a partially-learned model renders higher computation efficiency of short-term online adaptation. Existing solutions to online adaptation of OFDM neural channel estimators (Wang et al., 2022; Luan and Thompson, 2023; Zhang et al., 2023; Kong et al., 2025) still pose two challenges. First, conventional adaptation methods demand ground-truth channel coefficients as supervised labels, but such labels are unavailable online (Wang et al., 2022). Some studies (Luan and Thompson, 2023; Kong et al., 2025) obtain approximated channel coefficients using prior channel statistics, but acquiring such prior knowledge is also unrealistic. Second, how to avoid additional pilot overhead is crucial.

20011372

Existing work either doubles pilot overhead (Luan and Thompson, 2023) or periodically populates an entire OFDM frame with pilots for online adaptation (Wang et al., 2024, 2025a,b). Such schemes also incur control signaling overhead, as signaling messages are required to trigger transmission of these additional pilots (Luan and Thompson, 2023). To the best of our knowledge, there still lacks an online learning framework that can fully resolve the above two challenges. To this end, a *self-supervised task* that does not require true channel coefficients or additional pilots is introduced on top of the original channel-estimation task (i.e., *main task*). The key insight is that this new task must learn the latent features that benefit the main task. As a result, via online self-supervised learning (SSL) of this task alone, online features are captured and shared with the main task, thereby improving the main-task performance. This design rationale is inspired by test-time training (TTT) in machine learning (Sun et al., 2020; Gandelsman et al., 2022; Liu et al., 2021).

While designing the channel-estimation task is straightforward, there still exist three challenges: 1) how to design an appropriate self-supervised task (also named SSL task) that aligns with the above design principles; 2) how to design a model architecture to consolidate the SSL task with the original task; 3) how to design an effective and efficient training strategy.

Research Problem and Challenges of Long-Term Collaboration

Regarding long-term online generalization and robustness of neural receivers, single-BS learning is not sufficient. Rather, collaborative online retraining among multiple BSs is considered (Mashhadi et al., 2021). In a collaborative learning framework, each BS collaborates with other BSs to learn its neural receiver under the coordination of a central server, such that the knowledge accumulated in the entire multi-cell network can be exploited. Specifically, we study collaborative learning among multiple *fully-learned* neural receivers. Given that training can span a relatively large period, there is sufficient time and computational resources for each BS to train its fully-learned neural receiver effectively.

As shown in Fig. 1.5, multiple small BSs are connected to a central server via backhaul links (Wang et al., 2022), and the central server coordinates the process of multi-cell collaborative learning for uplink neural receivers. To assist online learning, each BS needs to build up a local labeled dataset based on pilot-based training sequences. Multi-BS collaborative learning holds two key advantages as compared to single-BS local learning. First, it overcomes the problem of limited observation of channel instances in local learning. Through collaborative learning, the neural receiver in a cell enriches its online knowledge by learning data distributions from other cells, so the model generalization ability is enhanced. Second, collaborative learning helps neural receivers improve their capability of handling inter-cell interference. As such interference encodes other cells'

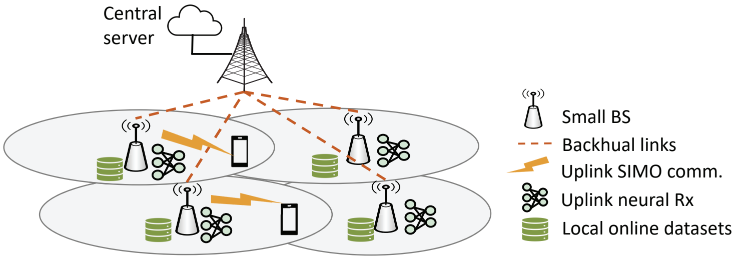


Figure 1.5: Multi-cell network with heterogeneous channel scenarios.

channel information, a neural receiver with multi-cell channel knowledge can better grasp the statistical features of interfered signals.

It is challenging to design an appropriate collaboration mechanism to fulfill online collaborative learning among multiple cells. A straightforward one is to consider centralized learning where multi-cell data are gathered in one server. However, this strategy is not always desirable for two main reasons. First, there exists a privacy concern (Niknam et al., 2020), as physical layer data at a BS are not supposed to be exposed to an entity in the cloud unless a cloud RAN is considered. Second, centralized learning demands a costly server with abundant storage and computing resources. Therefore, it is more desirable to develop a distributed collaborative learning scheme that keeps the data locally. A common framework that can be adopted is federated learning (FL) (McMahan et al., 2017; Mashhadi et al., 2021), where a global model is learned with locally-held data. However, due to data heterogeneity across cells, the unified global model may perform even worse than the local models, which indicates a lack of personalization.

1.3 Related Work

1.3.1 Relay Placement for Multi-Hop Wireless Networks

Relay placement for wireless multi-hop backhaul networks is a critical problem that attracts much attention from researchers (Hu and Blough, 2017; Chen et al., 2021; Ntontin et al., 2021). In (Hu and Blough, 2017), multiple relays are deployed between a pair of BSs in an mmWave backhaul network to overcome severe path loss and blockage issues. The placement of unmanned aerial vehicles (UAV) and reflective intelligent surfaces (RIS) for wireless backhauling are studied respectively in (Chen et al., 2021) and (Ntontin et al., 2021) to provide better alternative communication paths than the direct single-hop paths. However, the above work does not aim to boost network-wide throughput. The optimal placement of multiple RISs in RIS-aided indoor THz communications is studied in (Saqib et al., 2024). This scheme aims to maximize the coverage area and only two-hop links (i.e., BS-RIS-user links) are considered. In (Diamanti et al., 2021), a resource management

20011372

framework is designed for the IAB network. Although the framework can optimize the end-to-end data rate, only one IAB node and one IAB donor is considered in (Diamanti et al., 2021), and thus only a single two-hop backhaul link is formed. Such a system setup significantly simplifies the resource reuse scenario and directly eliminates the network bottleneck problem. A practical heuristic relay placement scheme is designed for multi-hop wireless networks in (Nikolov and Haas, 2016). It jointly optimizes relay node locations and traffic loads to minimize the overall number of packet retransmissions. By contrast, we aim to maximize the network saturation throughput. In (Minelli et al., 2014), a multi-stage relay placement scheme is designed to maximize network connectivity, where the coarse relay sites are identified first and then the ideal sites are selected and refined. However, the scheme is more focused on inter-node reachability, and it does not consider the flow distribution and flow fairness regarding network throughput. The relay placement or relay selection algorithms designed in (Magán-Carrión et al., 2016; Chang et al., 2014; Yang et al., 2012) aim to improve the throughput for relay-assisted cellular networks, but only two-hop transmission links between BS and users are taken into account.

Relay placement problem has also been studied in wireless sensor networks (Magán-Carrión et al., 2016; Liang et al., 2021), but their main objectives are to improve connectivity or reduce hop count for power-limited sensor nodes. Relay nodes are placed to maximize data throughput received at sink nodes in (Flushing and Caro, 2013). However, resource sharing and interference patterns are considered through the maximum number of flows that can circulate within a disk of a fixed radius centered at each node. This method cannot capture the interference pattern as accurately as the link conflict graph and cliques. Also, flow fairness is ignored in (Flushing and Caro, 2013).

Deep reinforcement learning (DRL) has been demonstrated to be a promising approach to solving sequential decision problems in many fields (Zhu et al., 2021; Ye et al., 2019; Shi et al., 2021; Liang et al., 2020; Zhao et al., 2019; Li et al., 2020). Considering a wireless backhaul network, Abdelmoaty et al. (2022) designs a multi-level hierarchical network architecture and determines which level each node is at to form the hierarchical topology based on DRL. There also exists some work focusing on the high-frequency wireless backhaul networks, such as mmWave backhaul. In (Huang and Wang, 2023), a Bayesian approach is designed to optimize the DAG structure of an integrated access and backhaul (IAB) network in mmWave bands, such that bursty traffic can be sustained with the highest probability. In (Simsek et al., 2021), a DRL-based topology formation approach is designed for mmWave IAB networks to maximize the minimum link capacities on the paths from IAB donors to IAB nodes. However, the above work does not consider the impact of inter-link interference, and their design objectives are not fundamentally providing more network capacity.

20011372

Compared to designing topologies based on the existing nodes, adding new nodes can provide much more opportunities for resolving network bottlenecks. Therefore, on the network infrastructure level, this dissertation is focused on the approach of relay placement to boost network throughput, and our novelty lies in the following aspects: 1) we consider a general multi-hop and multi-flow wireless backhaul network, where the flow fairness and flexible flow distribution are considered; 2) based on this set-up, we aim to maximize the fairness-based network saturation throughput that is not considered thoroughly before; 3) to achieve the throughput-maximization goal, the problem of relay placement is studied by considering the interactions among network bottlenecks for the first time.

1.3.2 Network Slicing for Radio Access Networks

Due to its capability to enable efficient sharing of wireless network resources in 5G systems, RAN slicing has attracted increasing attention in recent years(Tang et al., 2019; Mandelli et al., 2019; Guo and Suárez, 2019; Hua et al., 2020; Sun et al., 2021; D'Oro et al., 2018; Sexton et al., 2020; Ksentini and Nikaein, 2017). In (Ksentini and Nikaein, 2017), a programmable, 3GPP-compliant network slicing architecture is designed, which spans from CN to RAN. It is focused on the function block and architecture design. A slice-aware RAN framework proposed in (Guo and Suárez, 2019) includes a slice scheduler and a medium access control (MAC) scheduler per slice in each cell. This work is focused on the framework design rather than detailed scheduling schemes. In (D'Oro et al., 2018), RAN slicing is modeled as a congestion game and the slicing policy is determined for each slice in a distributed manner. In (Zanzi et al., 2021), a multi-armed-bandit-based (MAB) inter-slice scheduler allocates resources to multiple slices on a large timescale, which provides average latency and throughput guarantees for each slice in a single cell.

The above work is mainly focused on framework design and high-level scheduling. As a crucial step to map the logical planning of RAN slicing to the physical resources, the slice enforcement problem has also started to gain researchers' attention. In (Mandelli et al., 2019), frequency resources are allocated to the flows under the slice constraints in terms of aggregate rate targets. However, only a single base station is considered, and the algorithm cannot be directly extended to multi-cell scenarios. In (Tang et al., 2019), network-wide dedicated bandwidths are allocated to slices under two specific modes, eMBB and URLLC, to maximize the network operator's revenue while guaranteeing QoS satisfaction. This method assumes hard isolation among slices, and thus leaves a large space for spectral efficiency improvement. For multi-cell scenarios, a service provisioning framework is provided for RAN slicing (Sun et al., 2021), which jointly optimizes user admission and bandwidth allocation. However, this framework does not consider either the effect of slicing policies or slice isolation.

In (D'Oro et al., 2019), slice-level enforcement is conducted to mitigate inter-slice interference. It is the first work to ensure soft slice isolation. However, without link-level scheduling, this scheme cannot guarantee QoS performance of each link or adapt to practical unbalanced slicing policies.

DRL is also applied in the field of network slicing in recent years (Mei et al., 2021; Wu et al., 2021; Xiang et al., 2020; Dong et al., 2021; Addad et al., 2021; Hua et al., 2020; Chen et al., 2019; Liu et al., 2020; Azimi et al., 2022). A joint network slicing and routing mechanism is introduced in (Dong et al., 2021). This work leverages GCN-powered DRL to determine routing paths for each slice and conduct slice-level resource partitioning. In (Xiang et al., 2020), a joint content caching and UE mode selection problem is tackled via DRL, which is considered a critical problem in fog-RAN slicing. A slice-level resource allocation and workload distribution algorithm is designed in (Wu et al., 2021) for delay-sensitive and delay-tolerant vehicular services. It leverages two-layer constrained DRL to adapt to dynamic vehicle traffic density. A deep Q-network (DQN) is utilized to intelligently conduct slice-level scheduling in a single cell in (Hua et al., 2020).

Although most of the existing work is focused on slice-level scheduling, some work applies DRL to fine-grained slice enforcement. A hierarchical DRL framework is proposed to conduct slicing configuration adaption and user-level radio resource allocation in (Mei et al., 2021). Radio resources are allocated on a small timescale to optimize the QoS performance and spectrum efficiency in a single cell. In (Azimi et al., 2022), an energy-efficient resource allocation algorithm is designed to jointly optimize slicing policy and slice enforcement. RB resources are reserved for each slice on a large timescale, while slice admission, power, and RB allocation are determined on a small timescale. However, only hard isolation is considered in (Azimi et al., 2022), and there lacks interference management among links in both (Azimi et al., 2022) and (Mei et al., 2021). The channel scheduling among multiple slices is modeled as a stochastic game in (Chen et al., 2019) where A DRL-assisted online scheme is designed to conduct channel auction to maximize the network utility in multiple cells. Although user-level scheduling is realized, this work does not consider either long-term resource guarantee for each slice or users' QoS satisfaction.

In spite of the above related work, there lacks fine-grained link-level interference management in multi-cell RAN slicing. Such interference management is indispensable, as the existence of interference directly impacts slice isolation and users' QoS guarantee. Compared with the related work, our work is distinct in two aspects. First, a comprehensive framework is designed for multi-cell RAN slicing on two timescales. This framework jointly tackles long-term slicing policy conformance, short-term interference management, and link-level QoS guarantee. Second, interference management is conducted with link-level granularity, where link interference is captured via an interference-graph based GNN.

1.3.3 Design of Neural Receivers

Neural Channel Estimators

In the following, the approaches of DL-based channel estimation are reviewed first, followed by some research efforts on online adaptation of NN-aided channel estimators.

Neural channel estimators play a critical role in future wireless communications. Its objective is to learn the non-linear mapping from channel estimates at pilot positions to channel coefficients of an entire frame based on channel datasets. Such a data-driven channel estimator is first proposed in (Soltani et al., 2019, 2020), where ChannelNet consisting of the super-resolution and restoration convolutional neural networks (CNN) is employed to interpolate and denoise channel estimates. Residual learning is introduced to OFDM channel estimation in (Li et al., 2020) and the designed model ReEsNet is built on ResNet blocks (He et al., 2016). The above work is focused on CNN-based channel estimators. To improve model generalization ability and enable parallel computing, CNN is further replaced by the attention-based transformer in OFDM channel estimation (Vaswani et al., 2017; Luan and Thompson, 2022; Chen et al., 2020; Luan and Thompson, 2023). An end-to-end transformer model is designed in (Chen et al., 2020), but it incurs significant computation cost by applying attention on the non-pilot positions that are padded with zeros. To reduce computation cost, a hybrid architecture is proposed in (Luan and Thompson, 2022, 2023), consisting of the transformer, the fully-connected up-sampling layer, and the CNN.

To strike a balance between model performance and computation cost, our designed approach, ChannelMAE, adopts an MAE (He et al., 2022) architecture for the channel estimation task. This MAE has a transformer-based encoder and a fully-convolutional ResNet-based decoder. Although it is also a hybrid model of transformer and CNN, our designed model has two distinct advantages compared with the hybrid models in (Luan and Thompson, 2023). First, it is aware of the pilot layout by incorporating pilot-position information into the MAE decoder, such that it is more robust to variations in pilot layouts. Second, it features a fully-convolutional decoder and thus does not include any fully-connected layers, which makes the model adaptable to various OFDM frame size (i.e., number of OFDM symbols and subcarriers).

The above research efforts are focused on optimizing neural channel estimators offline, where ground-truth channel coefficients are used as labels for supervised learning. However, such labels are no longer available online. In this case, how to adapt a neural channel estimator online is still challenging. Recently, online adaptation of neural channel estimators has attracted growing research attention (Wang et al., 2022; Luan and Thompson, 2023; Zhang et al., 2023; Kong et al., 2025). In (Wang et al., 2022), a few-shot learning scheme is designed for channel estimators to fast adapt to new environments. However, it still assumes ground-truth

20011372

channel coefficients can be obtained. In (Luan and Thompson, 2023), a new class of pilot signals, termed label pilots, is introduced, and the channel estimates at these positions are treated as online training labels. Rather than predicting the entire OFDM channel, this scheme uses the channel estimates at the standard-aligned pilot positions as input and outputs the estimates at the label-pilot positions. These label-pilot estimates are then bilinearly interpolated to recover the channel response for the entire frame. For the label pilots, either the transmit power is substantially increased or prior channel statistics are assumed so that the resulting estimates can approximate the true channel responses. This approach also incurs a considerable additional pilot overhead. The online adaptation strategy in (Kong et al., 2025) adopts a similar methodology but focuses on continual online learning of channel estimators.

Beyond these studies, (Zhang et al., 2023) proposes an SSL technique that dispenses with ground-truth channel labels. It trains a denoising network (DnCNN) that takes received signals with additional synthetic noise as input and the original received signals as output, and the resulting model is applied directly to channel estimation. This method, however, is suitable only for narrow-band channels and cannot accommodate OFDM channels, where frequency selectivity must be considered. Extending the approach to OFDM requires first bilinearly interpolating the pilot-based channel estimates before passing them to DnCNN. Moreover, the CNN-based model used in (Zhang et al., 2023) has a lower computation efficiency than transformer-based alternatives.

Fully-Learned Neural Receivers

Compared with traditional model-based receivers, neural receivers can: 1) reduce the overheads incurred by reference signals to achieve a higher spectral efficiency (Aoudia and Hoydis, 2021); 2) adapt to highly-nonlinear real-world channels without knowing underlying channel models (Aoudia and Hoydis, 2022; Honkala et al., 2021); 3) learn to compensate for hardware-induced non-linear distortions (Pihlajasalo et al., 2023; Ye et al., 2017). There exist various architectures for neural receivers. As an early study in this field, the OFDM neural receiver is designed with a fully connected neural network in (Ye et al., 2017). Data-driven OFDM systems are further developed in (Ye et al., 2020, 2021) to jointly learn the transmitter and receiver. Furthermore, recurrent networks (Farsad and Goldsmith, 2018) and convolution networks (Honkala et al., 2021; Zhao et al., 2021) are also employed in neural receivers. Since the model architecture in (Honkala et al., 2021) matches the physical layer setup of this dissertation, it is adopted as a basic building block of neural receivers.

In parallel with the above work, some research efforts have been made to improve online performance of a neural receiver (Jiang et al., 2021; Park et al., 2021; Fischer et al., 2022). In (Jiang et al., 2021), an online model architecture named SwitchNet is proposed. In this architecture, the online model parameters consist

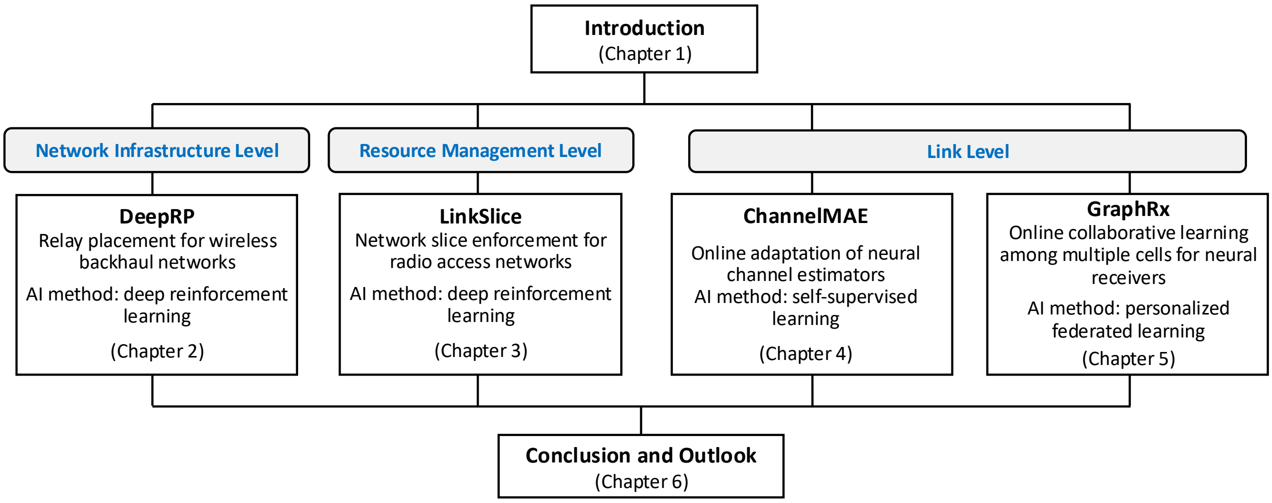

Introduction
(Chapter 1)
Network Infrastructure Level
Resource Management Level
Link Level
DeepRP
Relay placement for wireless backhaul networks
AI method: deep reinforcement learning
(Chapter 2)
LinkSlice
Network slice enforcement for radio access networks
AI method: deep reinforcement learning
(Chapter 3)
ChannelMAE
Online adaptation of neural channel estimators
AI method: self-supervised learning
(Chapter 4)
GraphRx
Online collaborative learning among multiple cells for neural receivers
AI method: personalized federated learning
(Chapter 5)
Conclusion and Outlook
(Chapter 6)

Figure 1.6: Organization of this dissertation.

of a weighted linear combination of all sub-network parameters, with only the weights being retrained online. This method implicitly assumes all online channels can be accurately modeled by linearly combining a limited set of sub-networks, which is an overly optimistic assumption in practice (Fischer et al., 2022). To facilitate on-the-fly retraining of neural receivers without ground-truth labels, online label recovery is designed via error-correction channel coding in (Fischer et al., 2022). Specifically, the corrected transmission bits are treated as pseudo-labels. However, this scheme suffers from erroneous labels under low SNRs. Meta-learning algorithms are employed in (Park et al., 2021) to enable fast adaptation to a new channel environment, but meta-learning-based models cannot retain offline environment knowledge after online adaptation. Furthermore, limited online channel knowledge at a single receiver restrains model generalization ability, which calls for collaborative learning among multiple receivers.

1.4 Developed Approaches and Organization of the Dissertation

To address the challenges stated in Section 1.2, this dissertation studies *AI-assisted designs at three different architectural levels*, namely network infrastructure level, resource management level, and link level. The organization of the dissertation is illustrated in Fig. 1.6, as elaborated in the following.

In Chapter 2, a DRL-based relay placement approach (*DeepRP*) is designed for this multi-step process of relay placement. To tackle the above three challenges, the key designs are carried out as follows. First, to capture the interactions among bottlenecks and determine the relationship between network architecture and network throughput, the bottleneck theory named the *clique-based bottleneck theory* is developed for general

20011372

wireless networks (with omnidirectional or directional links). The theory applies the notion of clique from graph theory to capture link conflict among wireless links and constructs a structure of bottlenecks in the unit of clique. With the clique-based bottleneck structure, the network throughput can be derived given a network architecture. Second, based on the constructed bottleneck structure, the clique gradient is derived to quantitatively determine how the perturbation of each bottleneck clique impacts the overall network throughput. Third, the relay placement problem is decomposed into a multi-step, iterative process where one relay is placed at each step, until the number of relays reaches the given requirement. *DeepRP* is designed for this multi-step process of relay placement. It is guided by the clique-based bottleneck theory, and featured with three key aspects: 1) the graph convolutional network (GCN) is employed to embed the node features in the network topology; 2) the clique gradients are used in the action masking scheme to guide the agent to select the sites in proximity to the most critical bottlenecks; 3) the clique-based bottleneck structure is constructed after each decision step, based on which the reward is computed. As a result, DeepRP can achieve much higher throughputs than the baselines.

In Chapter 3, a link-level slice enforcement scheme called *LinkSlice* is developed in this dissertation for a multi-cell network. It achieves fully soft, fine-grained slice isolation via link-level scheduling of PRBs for each slice. It satisfies QoS requirements demanded by different users. Moreover, LinkSlice cannot only ensure long-term (i.e., a number of radio frames) slicing policy conformance, but can also adapt flexibly to different patterns of slicing policy, either balanced or unbalanced. The design of LinkSlice is elaborated as follows. In a multi-cell RAN, a central controller collects channel conditions and user demands from all the BSs, and conducts link-level slice enforcement over multiple air interfaces. To determine PRBs for each slice at the link level, an optimization problem is formulated as minimizing the amount of allocated resources over each period of slicing policy, subject to the constraints that ensure soft slice isolation, QoS guarantee, and slicing policy conformance under link interference among different cells. The optimization problem is hard to solve due to extremely high computation complexity. However, by converting the constraint of slicing policy conformance into a penalty term of the objective function, the optimization problem is transformed into a certain type of optimization problem that can be effectively solved via a DRL-based approach (Zhu et al., 2021). By leveraging the DRL-based approach, LinkSlice is then designed as an iterative two-stage slice enforcement scheme. More specifically, in the first stage, transmission rates in communication links of different slices are determined via DRL. Interference among links is encoded as an interference graph, and then a graph convolutional network (GCN) is applied to capture the relationships among interfering links. A graph pooling mechanism is further considered to generate a compact and informative state space for the DRL agent. In the second stage, after

20011372

transmission rates are determined, PRB scheduling per radio frame is conducted for communication links of each slice. It can be solved by a heuristic algorithm called greedy slice enforcement with masking mechanism (G-SEM). The two stages work together iteratively until reaching convergence. By then, an effective link-level slice enforcement tool is obtained.

In Chapter 4, an SSL-assisted online adaptation framework for neural channel estimators, named *ChannelMAE*, is designed. First, the SSL task is designed as first randomly masking a large fraction of resource elements in each received frame and then reconstructing those masked parts, namely *masked reconstruction* (He et al., 2022). The reason for this design lies in the fact that received signals inherently carry information of channel coefficients. Therefore, reconstructing different masked parts of received signals implicitly learns channel statistical features in both time and frequency domains, and such features are also essential to the channel-estimation task. To enable effective reconstruction, this task must also incorporate the estimated data symbols as an input component besides the unmasked parts of the frame. These estimated symbols are obtained via an online symbol-recovery mechanism, where data-symbols are recovered through symbol detection online. Thus, no additional pilot overhead is incurred. Note that the recovered symbols cannot be used as pseudo-labels to back-propagate through the whole receiver to train the channel estimator, because the traditionally-designed detection modules can be non-differentiable (Raviv et al., 2023). In addition, the above task design results in high correlation between the SSL and main tasks, because channel estimation can be interpreted as a special case of masked reconstruction: it reconstructs channel coefficients from the noise-corrupted local observation at pre-known pilot positions. Such correlation leads to synergetic learning of the two tasks. Second, to consolidate the SSL task with the original task, a two-branch masked auto-encoder (MAE) model architecture called *ChannelMAE* is developed, with each branch dedicated to one task. The two branches share a feature *encoder* but have their task-specific *decoders*. The encoder is designed as a lightweight attention-based transformer (He et al., 2022), which produces low-dimensional latent representations from the input. Via the self-attention mechanism, it implicitly captures channel statistical features with high computation efficiency (Vaswani et al., 2017). Both decoders have fully-convolutional architectures but with different model sizes, performing reconstruction from the latent representations. Third, a two-phase training strategy is designed for ChannelMAE. In the offline pretraining phase, all model parameters are pretrained by optimizing the main and SSL branches together. In the online adaptation phase, only the shared encoder is updated online by optimizing the SSL branch. The online-channel features learned by the encoder are also used in the main branch, thereby improving online channel-estimation accuracy. In addition, it is crucial to reduce memory footprint during adaptation. Thus, batch-wise online learning is enforced, where each adaptation step is conducted over a batch of online streaming

20011372

samples that is immediately purged after use. To enrich data quantity and diversity, each online batch is further augmented by masking the same received frame with different random masks when optimizing the SSL task. With the above three key designs, our designed two-task learning framework holds several distinct features compared with the existing TTT framework (Sun et al., 2020): 1) TTT sets classification as its main task, whereas this chapter is focused on a regression problem; 2) TTT constructs a single-input multi-task problem by letting two tasks process the same input, whereas in our case the two tasks can have diverse inputs, resulting in a multi-input multi-task problem. 3) TTT performs per-sample adaptation before inference, while ChannelMAE enforces per-batch adaptation on a relatively larger timescale after inference of this batch, which is better suited to wireless communication applications.

In Chapter 5, a graph-based collaborative learning scheme, called *GraphRx*, is developed for uplink neural receivers to achieve a proper tradeoff between generalization and personalization. First, GraphRx is formulated as a personalized federated learning (PFL) problem, based on a weighted collaboration graph where a node on the graph denotes a personalized model at a BS while a weighted edge represents the collaboration intensity between a pair of models. Moreover, the graph weights and neural receiver models are jointly optimized so that generalization and personalization are optimized together. Second, to solve the problem, a two-step alternating method is applied: 1) update the personalized model at each BS, given a collaboration graph and an aggregated model sent from the server; 2) determine the collaboration graph at the server, given the updated personalized models from all BSs. The first step is simple, but there exists a challenge in the second step, as the server has no access to local datasets on BSs, which prevents it from conducting graph optimization. Thus, the graph optimization problem in the second step is transformed so that it does not rely on local data. The key idea is to derive a generalization bound of the collaborative learning problem, approximate it empirically, and then replace the objective function of collaborative learning by the approximate bound. In this way, the collaboration graph can be determined at the server without using the local data of BSs. As compared to the existing graph-based PFL algorithm (Ye et al., 2023), GraphRx is distinct in three aspects: 1) collaboration intensity is captured with finer granularity rather than just model cosine similarity; 2) no regularization with graph weights is necessary; 3) the collaboration graph is optimized by following the generalization bound of multi-source domain adaptation. To support online training, pilot-free methods in (Fischer et al., 2022) can be used where labels are directly retrieved from communication data, but their performance can be easily impacted by erroneous labels. Thus, pilot-based training sequences (i.e., *training pilots*) are adopted to retrain the neural receiver in this dissertation. To reduce overhead of training pilots, data augmentation is needed. (Fischer et al., 2022). Specifically, two techniques, rotation and noise injection, are employed at each BS to enrich the quantity

20011372

and diversity of online training data.

In Chapter 6, the conclusions of this dissertation are stated, and some interesting topics that worth future investigation are then elaborated.

20011372

Chapter 2

Relay Placement for Wireless Backhaul Networks Based on Reinforcement Learning

Backhaul mesh networks are critical for ensuring coverage and connectivity of high-frequency 6G networks. To maintain high throughput, its architecture needs to be augmented by adding relays. However, how to place relays at appropriate sites poses two challenges: 1) there lacks a theory to capture the relationship between a certain change of network architecture and its throughput gain; 2) selecting the best sites for relays is a complicated combinatorial problem. To tackle the first challenge, this chapter first establishes a clique-based bottleneck theory, through which a clique-based bottleneck structure of a given network architecture is constructed to determine the network throughput. Based on this bottleneck structure, clique gradients are then computed to quantify the impact of each clique on the overall network throughput. With the clique-based bottleneck theory, the second challenge is resolved by embedding clique gradients into a deep reinforcement learning (DRL) scheme. Specifically, the DRL actions are masked such that only the relay sites that match the highest clique gradients are selected. This DRL-based relay placement (DeepRP) scheme is evaluated via extensive simulations, and performance results show that it can boost network throughput by more than 50%, which is $10.4 - 32.1\%$ higher than those of baseline schemes.

2.1 System Model

2.1.1 Network Model

As shown in Fig. 1.1, a TDD-based backhaul mesh network with one mesh gateway and multiple deployed mesh nodes is considered. There also exist multiple known candidate relay sites. The relays to be deployed are dedicated relays as in Hu and Blough (2017), which do not serve as access points for users. The mesh gateway can gather network-wide information and determine the sites for relay placement in a centralized way. As the network is assumed to operate in high-frequency bands (e.g., THz bands), directional links are used among all the network nodes. Let $\mathcal{N}$, $\mathcal{L}$, $\mathcal{F}$ be the set of network nodes, directed links, and flows, and n, l, f be the indices that refer to the entities, respectively. The capacity of link l is denoted as C_l. The resources allocated to each flow is represented in data rates (bps) instead of the physical time and frequency resources, which will be elaborated in Section 2.2. Note that the feasibility on enforcing data rates into physical resources with a certain MAC mechanism is outside the scope of this chapter. This issue is addressed in Li and Patras (2020); Huang and Bensaou (2001); Jain et al. (2003).

There are multiple end-to-end aggregate flows in the backhaul network, either between the mesh node and the mesh gateway, or between two mesh nodes. The single-path routing scheme is adopted, and $\mathcal{D} = \{\mathcal{R}_1, ..., \mathcal{R}_F\}$ is the set of routes with $\mathcal{R}_f$ as flow f's routes. $\mathcal{D}$ is also referred to as *flow distribution.* A stable flow distribution is assumed to be known at the beginning of relay placement (Li and Patras, 2020). For backhaul networks, the source and destination of each aggregate flow change on a large timescale, which can be obtained by network operators via traffic monitoring (Chao and Hsu, 2023). During one procedure of relay placement, the source-destination pairs stay unchanged (Li and Patras, 2020), while the routing paths can be flexibly adjusted.

A three-sector analog beamforming architecture is adopted, and beam switching is applied for each sector (only one link can be activated at one time in the same sector). The effective gain including antenna gain and beamforming gain within the main-lobe beam is assumed to be constant as in Saha and Dhillon (2019). Let G_t and G_r be the transmit effective gain and receive effective gain, and perfect beam alignment is assumed. To model multi-path channel components, the Saleh-Valenzuela (S-V) model adopted (Lin and Li, 2015; Cho et al., 2010). We only consider one ray within each cluster (Cho et al., 2010), as the finer details of each cluster's internal multi-path components are not critical in our problem. Therefore, the channel impulse response of the S-V model is expressed as

$$h(t) = \sqrt{L\left(f_c, d\right)} X \sum_{m=0}^{M} \beta_m e^{j\theta_m} \delta\left(t - T_m\right), \tag{2.1}$$

where $L\left(f_c, d\right)$ is the path loss coefficient that will be elaborated later; X is a log-normal random variable

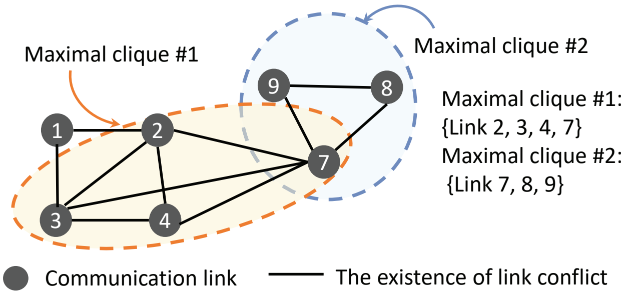


Figure 2.1: Link conflict graphs and maximal cliques.

with the standard deviation as σ_X for the shadowing effect; M is the number of non-negligible clusters within time duration T_s; θ_m is a random variable uniformly distributed over $[0, 2\pi)$; T_m is the arrival time of the m-th cluster modeled by a Poisson process with an average arrival rate of λ; β_m follows the Rayleigh distribution $f(\beta_m) = 2(\beta_m/\overline{\beta_m^2})e^{-\beta_m/\overline{\beta_m^2}}$ with $\overline{\beta_m^2}$ is the average power in the m -th cluster. The average power is given as $\overline{\beta_m^2} = \overline{\beta_0^2}e^{-T_m/\gamma}$ with γ as the cluster decay constant and $\overline{\beta_0^2} = 1$.

The path loss coefficient $L(f_c, d)$ includes the free-space spreading loss $L_s(f_c, d)$ and the molecular absorption loss $L_{abs}(f_c, d)$ in THz bands (Lin and Li, 2015), expressed as

$$\begin{aligned} L(f_c, d) &= L_s(f, d)L_{abs}(f, d) \\ &= G_r G_t \left(\frac{c}{4\pi f_c d_0}\right)^2 \left(\frac{d_0}{d}\right)^n e^{-k_{abs}(f_c)d}, \end{aligned} \tag{2.2}$$

where c is the speed of light; f_c is the carrier frequency; d is the propagation distance; d_0 is the reference distance set as 1; $k_{abs}(f_c)$ is frequency-dependent absorption coefficient; n is the path loss exponent.

2.1.2 Link Conflict Graph and Maximal Clique

The undirected link conflict graph (Wang et al., 2022) $\mathcal{G}_c = \langle \mathcal{V}_c, \mathcal{E}_c \rangle$ captures resource conflicts among links, where the vertex set $\mathcal{V}_c$ consists of all the transmission links, and the edge set $\mathcal{E}_c$ describes the conflicting relationship of wireless resources between two transmission links. Note that the transmission link from node A to B is considered as a different link than the transmission link from node B to A. Let $e_{ij} \in \mathcal{E}_c$ be the edge between vertex i and vertex j. The edge exists if 1) there is cross-link interference between two links (i.e., the receiver of one link is within the beam interference range of another link), or 2) the two links cannot be active simultaneously due to the TDD constraint, or 3) beam switching is applied between the two links in the same sector. The two links cannot use the same resources if there exists an edge between them.

Based on the link conflict graph, the notion of *clique* is utilized as a critical tool for capturing the interference

20011372

and resource sharing pattern. Each clique is a complete subgraph of the link conflict graph, and a *maximal clique* is the clique that is not contained in any other cliques. Via the existing algorithm (Cheng et al., 2011) of solving maximal clique cover of a graph, a set of maximal cliques $\mathcal{K}$ is obtained over the link conflict graph $\mathcal{G}_\text{c}$. Only one link in a maximal clique can be activated in one resource unit.

2.2 Clique-Based Bottleneck Theory

The clique-based bottleneck theory is established. In this theory, the clique-based bottleneck structure is constructed to determine the relationship between the network architecture and the network throughput. Based on the constructed structure, the clique gradients are computed to quantify the impact of each bottleneck clique on the throughput.

2.2.1 Clique-Based Bottleneck Structure

The bottleneck structures are constructed in the unit of maximal cliques to capture the link resource conflicts in wireless networks. To achieve this, the definition of a bottleneck clique is given, and then the algorithm of constructing the structure is developed based on clique fair shares.

Definition of Bottleneck Clique

The resource share of flow rate r_f on link l is defined as r_f/C_l. Since any two links within a maximal clique cannot be activated in the same resource unit, the sum of resource shares within each clique cannot exceed one. Therefore, the resource sharing pattern within each clique must satisfy the following feasibility constraint (Wang et al., 2008): $\sum_{l\in\mathcal{L}_k}\sum_{f\in\mathcal{F}_l} r_f/C_l \leq 1, \forall k = 1, 2, ..., K$, where $\mathcal{L}_k$ is the set of links in clique k, and $\mathcal{F}_l$ is the set of flows traversing link $l \in \mathcal{L}_k$. When the above feasibility constraint is satisfied, how to further enforce the flow rates into physical resources by MAC mechanisms is another critical problem that has been addressed in Li and Patras (2020); Wang et al. (2008). While the bottleneck clique can be defined under various fairness criteria (Ros-Giralt et al., 2022), this chapter is focused on the widely-used max-min fairness among end-to-end flow rates. The definition of bottleneck clique under max-min fairness extends the concept in Huang and Bensaou (2001); Wang et al. (2008) for multi-hop flows, which is stated as follows.

Definition 2.1 (Bottleneck Clique)**.** Clique k is a bottleneck clique for flow f if and only if the following conditions are satisfied: a) clique k must be a saturated clique that satisfies the feasibility constraint and has no remaining resource share for any flow, expressed as $\sum_{l\in\mathcal{L}_k}\sum_{f\in\mathcal{F}_l}\frac{r_f}{C_l} = 1$; b) the flow f's resource share under

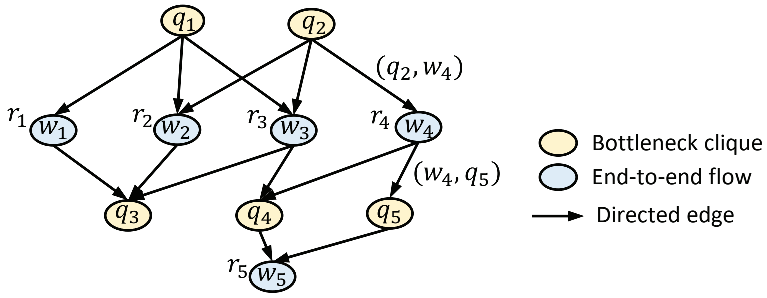


Figure 2.2: Clique-based bottleneck structure.

max-min fairness cannot be smaller than the fair shares of other flows on each link in clique k, expressed as $\frac{r_f}{C_l} \geq \frac{r_{f'}}{C_l}, \forall l \in \mathcal{L}_k, f' \in \mathcal{F}_l \setminus \{f\}$.

The above definition indicates that if the flow can only obtain a resource share smaller than the fair share obtained by other flows in clique k, this flow must be bottlenecked elsewhere other than clique k.

Algorithm of Constructing Clique-based Bottleneck Structures

From the above definition, multiple bottleneck cliques can be identified if the max-min fair flow rates $\{r_f | f \in \mathcal{F}\}$ are determined. However, there exist two questions: a) how to determine the max-min flow rate allocation for a backhaul mesh network based on maximal cliques; b) how the identified bottleneck cliques exert influence on each other. These two questions are merged and resolved by the construction of clique-based bottleneck structures. With the constructed bottleneck structure, the interactions among bottlenecks are captured, and the relationship between the network architecture and the network throughput is determined.

Before developing the algorithm for constructing a clique-based bottleneck structure, a critical metric named *clique fair share* s_k is introduced for clique k to specify the fair rate that each unconverged flow can be allocated in this clique. A flow is considered unconverged if it has not been allocated with the data rate of its fair share. Let $\mathcal{F}_l^{\mathrm{c}}$ and $\mathcal{F}_l^{\mathrm{u}}$ denote the set of converged flows on link l and the set of unconverged flows on link l respectively, and $\mathcal{F}_l^{\mathrm{u}} = \mathcal{F}_l \setminus \mathcal{F}_l^{\mathrm{c}}$. The definition of clique fair share is stated as follows.

Definition 2.2 (Clique Fair Share)**.** The clique fair share for clique k is obtained by equally allocating the remaining resource share among all the unconverged flows in this clique, expressed as

$$s_k = \frac{1 - \sum_{l \in \mathcal{L}_k} \sum_{f \in \mathcal{F}_l^{\mathrm{c}}} (r_f / C_l)}{\sum_{l \in \mathcal{L}_k} (|\mathcal{F}_l^{\mathrm{u}}| / C_l)}, \tag{2.3}$$

where the remaining resource share is obtained by deducting the converged flows' shares from 1 as shown in the nominator.

20011372

Algorithm 1 Bottleneck structure construction

```
1: **Input:** The set of maximal cliques $\mathcal{K}$, the set of flows $\mathcal{F}$;
2: **Initialization:** $\mathcal{V}_\mathrm{b} = \mathcal{E}_\mathrm{b} = \mathcal{F}^\mathrm{C} = \emptyset$, $r_f = 0, \forall f \in \mathcal{F}$, $\hat{\mathcal{F}}_k$ (the flow set traversing clique $k$), $\mathcal{Q}_f$(the set of cliques that are traversed by flow $f$), $\mathcal{F}^\mathrm{C}$(the converged flow set).
3: **repeat**
4:    Compute clique fair shares $s_k, \forall k \notin \mathcal{V}_\mathrm{b}$;
5:    $m_k = \min\left\{s_j | \hat{\mathcal{F}}_j \cap \hat{\mathcal{F}}_k \neq \{\emptyset\}, \forall j \notin \mathcal{V}_\mathrm{b}\right\}, \forall k \notin \mathcal{V}_\mathrm{b}$;
6:    **for** $k$ with $s_k = m_k$ **do**
7:       **for** $f \in \hat{\mathcal{F}}_k, f \notin \mathcal{F}^\mathrm{C}$ **do**
8:          $r_f = s_k$;
9:          $\mathcal{F}^\mathrm{C} = \mathcal{F}^\mathrm{C} \cup \{f\}$;
10:         $\mathcal{E}_\mathrm{b} = \mathcal{E}_\mathrm{b} \cup \{(\mathrm{q}_k, \mathrm{w}_f)\}$;
11:         $\mathcal{E}_\mathrm{b} = \mathcal{E}_\mathrm{b} \cup \{(\mathrm{w}_f, \mathrm{q}_j)\}, \forall j \in \mathcal{Q}_f \setminus \{k\}$;
12:      **end for**
13:      $\hat{\mathcal{Q}} = \hat{\mathcal{Q}} \cup \{k\}$;
14:      $\mathcal{V}_\mathrm{b} = \hat{\mathcal{Q}} \cup \mathcal{F}^\mathrm{C}$;
15:   **end for**
16: **until** $\mathcal{F}^\mathrm{C} = \mathcal{F}$
17: Obtain the network throughput $T = \sum_{f \in \mathcal{F}} r_f$;
18: **Output:** $\mathcal{G}_\mathrm{b} = \langle \mathcal{V}_\mathrm{b}, \mathcal{E}_\mathrm{b} \rangle, T$.
```

Based on the definition of clique fair share, Algorithm 1 is developed to construct a clique-based bottleneck structure, i.e., a DAG $\mathcal{G}_\mathrm{b} = \langle \mathcal{V}_\mathrm{b}, \mathcal{E}_\mathrm{b} \rangle$. As shown in Fig. 2.2, the vertex set $\mathcal{V}_\mathrm{b}$ consists of all the bottleneck cliques $\hat{\mathcal{Q}}$ and flows $\mathcal{F}$ in the network (i.e., $\mathcal{V}_\mathrm{b} = \hat{\mathcal{Q}} \cup \mathcal{F}$), while the edge set $\mathcal{E}_\mathrm{b}$ comprises of the directed edges between bottleneck cliques and flows. Let $(\mathrm{q}_k, \mathrm{w}_f)$ and $(\mathrm{w}_f, \mathrm{q}_k)$ denote the directed edge from clique k to flow f, and the directed edge from flow f to clique k, respectively, where $k \in \hat{\mathcal{Q}}, f \in \mathcal{F}$. The main procedure of Algorithm 1 is described as follows. In each iteration, the cliques that are not bottlenecks with the minimum clique fair share value are chosen (Line 4-5), e.g., q_1 and q_2 are chosen and added into the bottleneck structure in Fig. 2.2. Next, the flows traversing the chosen cliques are allocated with the clique fair share as their data rates, and they are added into $\mathcal{G}_\mathrm{b}$ with the directed edges from the cliques (Line 8-10). For example, w_4 traverses w_2, so the edge $(\mathrm{q}_2, \mathrm{w}_4)$ is added. Furthermore, the edges are added from these flows to other bottleneck cliques they traverse (Line 11), such as $(\mathrm{w}_4, \mathrm{q}_5)$. After the edge addition, the clique fair share values are recomputed by excluding the converged flows (Line 4). The iteration continues until all the flows are converged, and thus $\mathcal{G}_\mathrm{b}$ is established.

The time complexity of Algorithm 1 is analyzed as follows. Only Line 5 runs in non-constant time while other lines of statements run in constant time. In Line 5, the sorting based on Min-Heap is adopted with the heap size set to K at most, where K is the number of cliques. Each heap update runs in $O(\log K)$. The number of such updates is bounded by $K\eta$, where η is the maximum number of flows traversing a clique. Therefore, the algorithm has the complexity of $O(\eta K \log K)$.

20011372

The clique-based bottleneck structure is a non-trivial extension of the original link-based bottleneck structure in Ros-Giralt et al. (2022) Similar to the original structure, the clique-based structure captures the interactions among bottlenecks and flows via directed edges. However, there exists one distinct feature in the clique-based case: the bottleneck cliques can be overlapped by including some common links. Therefore, the theorem in Ros-Giralt et al. (2022) saying that the root vertices in the bottleneck structure have a higher impact on the overall network than the leaf vertices cannot be applied to the clique-based case. To accurately quantify the impact of each bottleneck cliques on the overall network, clique gradients need to be derived, as shown in the following subsection.

2.2.2 Clique Gradient

Definition of Clique Equivalent Capacity

To quantitatively characterize the capability of a clique, the clique equivalent capacity is defined via a set of virtual scheduling variables.

Definition 2.3 (Clique Equivalent Capacity)**.** The clique equivalent capacity R_k is defined as the weighted sum of all the link capacities in clique k, $R_k = \sum_{l \in \mathcal{L}_k} p_{k,l} C_l, \forall k \in \mathcal{K}$, where $\{p_{k,l} | l \in \mathcal{L}_k\}$ are the virtual scheduling variables that satisfy two constraints: $\sum_{l \in \mathcal{L}_k} p_{k,l} \leq 1, \forall k \in \mathcal{K}$; $p_{k,l} \geq 0, \forall l \in \mathcal{L}_k, k \in \mathcal{K}$.

With the virtual scheduling variables $\{p_{k,l} | l \in \mathcal{L}_k\}$, the capacity of a clique is characterized based on its link capacities and flow distribution. Analogous to how a link capacity is allocated among flows, the clique equivalent capacity is allocated among its traversing flows, from which the equivalent set of max-min flow rates as in Algorithm 1 can be obtained, as stated in the following theorem.

Theorem 2.1. *There exists a unique solution of* $\{p_{k,l} | l \in \mathcal{L}_k\}$ *for each clique in the bottleneck structure that achieves the same flow rate results as those obtained by Algorithm 1.*

The proof is given in Appendix A.

Derivation of Clique Gradients

The clique gradient $\mathbf{g}(j)$ is defined as the impact of an infinitely small perturbation in clique j's equivalent capacity on the overall network throughput T, expressed by

$$\mathbf{g}(j) = \frac{\partial T}{\partial R_j} = \frac{\partial \sum_{f \in \mathcal{F}} r_f}{\partial R_j} = \sum_{f \in \mathcal{F}} \frac{\partial r_f}{\partial R_j}. \tag{2.4}$$

From the above expression, $\mathbf{g}(j)$ can be also viewed as the sum of the clique j's impacts on all the flow rates. The key question is how to efficiently compute $\mathbf{g}(j)$ for each bottleneck clique j based on the constructed bottleneck structure $\mathcal{G}_{\mathrm{b}}$. Assume that an infinitely small perturbation is imposed on clique j (viewed as the perturbation source), i.e., ΔR_j is exerted on R_j. For a bottleneck structure $\mathcal{G}_{\mathrm{b}}$, let Δs_k be the perturbation on the clique fair share of bottleneck clique k, and Δr_f be the perturbation on the rate of flow f, both of which are caused by the perturbation source ΔR_j. For each directed edge in $\mathcal{G}_{\mathrm{b}}$, the former(latter) vertex is called the predecessor(successor) of the latter(former) one. Note that Δs_k and Δr_f are normalized with ΔR_j to remove the physical dimension, and thus ΔR_j is not included in the propagation. To figure out how this perturbation ΔR_j propagates through the whole bottleneck structure and results in Δs_k and Δr_f, the propagation equations must be derived.

First, the propagation equation for Δs_k is determined as follows. The variation of the remaining resource in clique k consists of two parts: a) the variation of capacity caused by the perturbation source, expressed as the ratio of common virtual scheduling variables in two cliques k and j, i.e., $\left(\sum_{l\in\mathcal{L}_k\cap\mathcal{L}_j} p_{k,l}\right) / \left(\sum_{l\in\mathcal{L}_j} p_{j,l}\right)$; b) the variation of consumed resources caused by Δr_f from all the predecessor flows of clique k, expressed as $-\sum_{f\in\mathcal{P}_k}\sum_{l\in\mathcal{L}_k\cap\mathcal{R}_f}(p_{k,l}\Delta r_f)$. By dividing the variation of remaining resources among all the successor flows of this clique, Δs_k can be derived, as shown in the following propagation equation:

$$\Delta s_k = \frac{\frac{\sum_{l\in\mathcal{L}_k\cap\mathcal{L}_j} p_{k,l}}{\sum_{l\in\mathcal{L}_j} p_{j,l}} - \sum_{f\in\mathcal{P}_k}\sum_{l\in\mathcal{L}_k\cap\mathcal{R}_f}(p_{k,l}\Delta r_f)}{|\mathcal{B}_k|}, \tag{2.5}$$

where $\mathcal{P}_k$ and $\mathcal{B}_k$ represent the predecessor and the successor vertices of clique k in $\mathcal{G}_{\mathrm{b}}$.

Second, the propagation equation for Δr_f is obtained by taking the minimum of the clique fair share perturbations on flow f's predecessor cliques, as the flow is bottlenecked by the clique with the minimum fair share. Therefore, the perturbation on the flow f's rate is

$$\Delta r_f = \min_{k\in\hat{\mathcal{P}}_f} \Delta s_k, \tag{2.6}$$

where $\hat{\mathcal{P}}_f$ represents the predecessor vertices of flow f in $\mathcal{G}_{\mathrm{b}}$.

After two perturbation equations are derived, the algorithm of computing clique gradients named *Clique-Grad* can be developed: Starting from the perturbation source clique j, the two equations (2.5) and (2.6) are sequentially computed to obtain Δs_k for each clique and Δr_f for each flow in the bottleneck structure. The sequence of propagation is the sequence by which the clique or flow is added into $\mathcal{G}_{\mathrm{b}}$ in Algorithm 1. The time complexity of CliqueGrad is analyzed as follows. The two propagation equations, Eq. (2.5) and Eq. (2.6), run in constant time. To propagate the perturbation from the roots of the bottleneck structure to the leaves, a Min-Heap of size KF is built as in Ros-Giralt et al. (2022), where F is the number of flows. The operation of heap updates runs at most KF times (Ros-Giralt et al., 2022), leading to $O(KF \log(KF))$.

It can be proved that the output of CliqueGrad by taking the sum of $\Delta r_f (\forall f \in \mathcal{F})$ is equal to the clique j's gradient, as stated in the following theorem.

Theorem 2.2 (The correctness of clique gradient computation)**.** *Assume that an infinitely small perturbation is imposed on clique j. By executing CliqueGrad, Δs_k and Δr_f can be achieved as*

$$\Delta s_k = \frac{\partial s_k}{\partial R_j}, \forall k \in \hat{\mathcal{Q}}, \tag{2.7}$$

$$\Delta r_f = \frac{\partial r_f}{\partial R_j}, \forall f \in \mathcal{F}. \tag{2.8}$$

As a result, the algorithm output $\hat{\mathbf{g}}(j)$ computed as $\sum_{f \in \mathcal{F}} \Delta r_f$ is equal to the clique gradient $\mathbf{g}(j)$.

The proof is given in Appendix A.

2.3 DRL-based Relay Placement with Clique-based Bottleneck Theory

2.3.1 Relay Placement Problem

The problem of relay placement is placing a set of relay nodes $\hat{\mathcal{R}}$ at the selected candidate relay location sites to boost the network throughput. Let $\hat{\mathcal{R}}_{\text{can}}$ be the set of all the candidate sites, and N_{r} be the number of relays that can be deployed considering the total deployment cost. $\hat{\mathcal{R}}_{\text{can}}$ and N_{r} are considered as given information, and $\hat{\mathcal{R}} \in \hat{\mathcal{R}}_{\text{can}}, |\hat{\mathcal{R}}| = N_{\text{r}}$.

With the clique-based bottleneck theory developed in Section 2.2, two critical theoretical results have been drawn: 1) by constructing the bottleneck structure, the network throughput is determined given a network architecture as in Algorithm 1; 2) the impact of each bottleneck clique on the overall throughput, i.e., the clique gradient, is determined based on CliqueGrad. However, even with these theoretical tools, it is still challenging to solve the relay placement problem. This is because the objective of relay placement in this chapter is unique: it aims to maximize the fairness-based network saturation throughput. To derive such fairness-based network throughput, one must rely on the algorithm of bottleneck structure construction (Algorithm 1) developed in

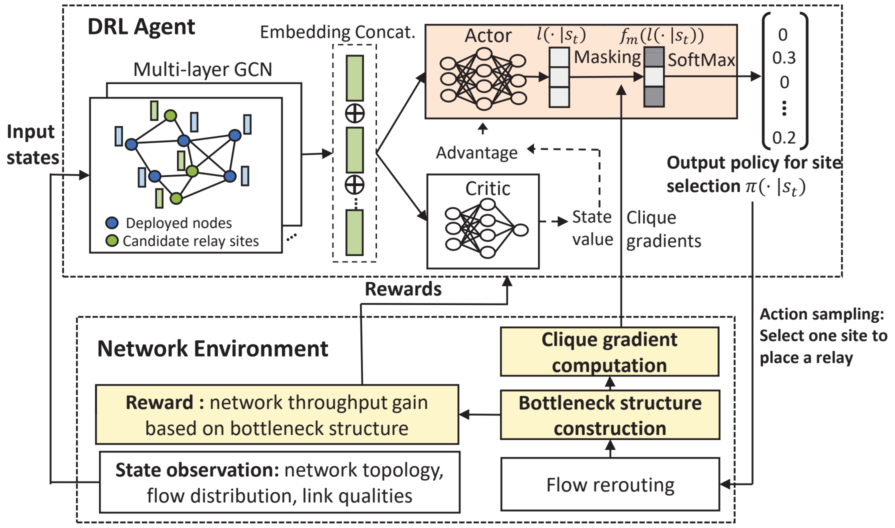

DRL Agent
Embedding Concat.
Multi-layer GCN
Input states
Deployed nodes
Candidate relay sites
Actor
$l(\cdot|s_t)$
$f_m(l(\cdot|s_t))$
Masking
SoftMax
0
0.3
0
⋮
0.2
Output policy for site selection $\pi(\cdot|s_t)$
Advantage
Critic
State value
Clique gradients
Rewards
Network Environment
Clique gradient computation
Reward : network throughput gain based on bottleneck structure
Bottleneck structure construction
State observation: network topology, flow distribution, link qualities
Flow rerouting
Action sampling: Select one site to place a relay

Figure 2.3: DeepRP: DRL Design Framework

the bottleneck theory. However, the derivation of throughput via Algorithm 1 cannot be analytically expressed as a differentiable function to be optimized. Therefore, this problem does not fall into traditional integer programming (Belotti et al., 2013), and thus the traditional optimization techniques such as branch-and-bound lose their effectiveness here (Belotti et al., 2013). This motivates us to develop a DRL-based method that does not require the objective function to be differentiable.

Instead of selecting multiple candidate sites at one time that incurs high complexity in DRL search space, we decompose the relay placement problem into *a multi-step, iterative process* where one relay is deployed at each step. In each step, a clique-based bottleneck structure is first constructed and then the clique gradients are computed based on the structure. The cliques with H highest gradients are named as *H-highest-gradient cliques*, which are identified as *high-impact cliques* for the network. The best candidate relay site that is in proximity of these cliques is selected. After deploying the new relay at the selected site by DRL, the flow distribution is changed accordingly by rerouting, and the new network architecture is obtained. The bottleneck structure can be then constructed again to start another iteration. The above DRL-based approach aims to maximize network throughput over multi-step decisions.

2.3.2 Key Designs of DeepRP

As shown in Fig. 2.3, the DRL framework named Deep Relay Placement (*DeepRP*) is designed based on an actor-critic architecture as elaborated in Section 2.3.3. The mesh gateway is considered as the centralized DRL agent, which selects the action a_t based on the current state $\mathbf{s}_t$ at step t. Then, the agent collects the reward r_t from the environment and updates the learnable parameters of DRL agent. Before elaboration on key modules, the state, action, and reward are described as follows.

State Representation. Since the information on candidate relay sites is known by the agent, a full virtual network topology $\mathcal{G}_\mathrm{w} = (\mathcal{V}_\mathrm{w}, \mathcal{E}_\mathrm{w})$ is constructed, where the node set $\mathcal{V}_\mathrm{w}$ includes all the deployed mesh nodes and candidate relay sites, and the edge set $\mathcal{E}_\mathrm{w}$ includes all the potential communication links among nodes. $\mathcal{G}_\mathrm{w}$ is considered as an undirected graph, and the edge weight is the estimated SNR of each undirected link normalized into the range of $(0, 1]$. Each node's feature has four elements: whether this node is deployed or not (represented by a binary value), the number of incoming flows, the number of outgoing flows, and the connectivity degree of this node in $\mathcal{G}_\mathrm{w}$. Let $\mathbf{F}$ denote the matrix of node features with the size $|\mathcal{V}_\mathrm{w}| \times 4$. Thus, the state is represented by $\mathbf{s}_t = \{\mathcal{G}_\mathrm{w}, \mathbf{F}\}$.

Action Representation. The action a_t is selecting one site among all the candidate sites to place a new relay. As a policy-based DRL method is adopted (Schulman et al., 2017), a_t is sampled based on a probability distribution vector generated by the DRL actor network (i.e., a *DRL policy* $\pi(\cdot|s_t)$). Specifically, the policy vector has the dimension of the number of candidate sites $|\hat{\mathcal{R}}_\mathrm{can}|$, and each element of this vector is the probability of choosing a specific site for relay deployment.

Reward Function. The reward assigned to step t is the network throughput gain after deploying the new relay, i.e., $r(\mathbf{s}_t, a_t) = T_{t+1} - T_t$, where T_{t+1} is obtained under the new network architecture after action a_t is taken. By computing losses from rewards, the following learnable networks are updated: GCNs $\boldsymbol{\theta}_\mathrm{g}$, actor network $\boldsymbol{\theta}_\mathrm{A}$ that generates policy $\pi(\cdot|s_t)$), and critic network $\boldsymbol{\theta}_\mathrm{V}$ that outputs a scalar $(V^\pi(\mathbf{s}))$ based on the current state. The details of loss computation and updating process are in Section 2.3.3.

As shown in Fig. 2.3, three key modules are designed for DeepRP to efficiently resolve the relay placement problem: 1) GCN-based graph embedding; 2) clique-gradient-based action masking; 3) bottleneck-structure-based reward computation. In the following, these module designs are elaborated.

GCN-Based Graph Embedding. The DRL agent conducts node embedding over the full virtual network topology $\mathcal{G}_\mathrm{w}$. Let A be the adjacency weight matrix of $\mathcal{G}_\mathrm{w}$ with self-connections and D is a diagonal matrix whose diagonal elements are $D_{ii} = \sum_j A_{ij}$. Considering a multi-layer GCN, the layer-wise propagation rule [1] for layer ℓ to obtain the $(\ell+1)$-th layer embedding matrix $X^{(\ell+1)}$ is

$$X^{(\ell+1)} = \sigma\left(D^{-1/2} A D^{-1/2} X^{(\ell)} \boldsymbol{\theta}_\mathrm{g}^{(\ell)}\right), \tag{2.9}$$

where $X^{(0)} = \mathbf{F}$ is the input node feature matrix, and $\boldsymbol{\theta}_\mathrm{g}^{(\ell)}$ denotes the parameters of ℓ-th layer GCN, and $\sigma(\cdot)$ is the ReLU activation function. A single-layer GCN aggregates one-hop neighbor nodes' states for each node, and thus each node can obtain a node-level embedding vector that encodes multi-hop node states after multi-layer GCN embedding. As shown in Fig. 4, a graph-level embedding is then obtained by concatenating all the embeddings of candidate relay nodes, and its dimension is $|\hat{\mathcal{R}}_{\text{can}}|d_x$ with d_{x} as the dimension of one node embedding. Note that $|\hat{\mathcal{R}}_{\text{can}}|$ is the number of candidate sites. This concatenated embedding serves as the input for both the actor and critic networks subsequently.

Clique-Gradient-Based Action Masking. To generate a DRL policy that enables efficient learning, the logit-based action masking (Huang and Ontañón, 2020) is designed based on clique gradients, which includes three steps. First, a set of valid actions $\mathcal{A}_{\text{valid}}$ is defined: the action of selecting an undeployed candidate site that is in proximity to H-highest-gradient cliques is considered valid. More specifically, a site is in proximity of one clique if it is in one-hop communication range of at least two nodes in the clique. In this way, the range of site selection is significantly narrowed down to be aimed at resolving the high-impact bottlenecks. In addition, if a candidate site is already selected in previous steps, it cannot be selected again, and it is excluded from the valid action set. Second, a masking function $f_m(\cdot)$ is designed, and it is applied on the logits, $l\left(\cdot|s_t\right)$, generated by the last layer of actor network $\boldsymbol{\theta}_{\text{A}}$. Let $M = |\hat{\mathcal{R}}_{\text{can}}|$ for ease of notation, and let a^i denote action i is taken based on s_t. The masking function is

$$f_m\left(l\left(a^i|s\right)\right) = \begin{cases} l\left(a^i|s\right), & \text{if } a^i \in A_{\text{valid}} \\ \eta, & \text{if } a^i \notin A_{\text{valid}} \end{cases}, \tag{2.10}$$

where $i = 1, \ldots, M$, and the invalid actions are masked out by setting their logits to a large negative number $\eta = -1 \times 10^8$ (Huang and Ontañón, 2020). Third, Softmax activation is applied on the masked logits to generate the final probability distribution over all the actions, i.e., DRL policy $\pi(\cdot|s)$, expressed by

$$\begin{aligned} \pi\left(\cdot|s_t\right) &= \left[\pi\left(a^1|s_t\right), \ldots, \pi\left(a^M|s_t\right)\right] \\ &= \text{softmax}\left(f_m\left(l\left(a^1|s_t\right)\right), \ldots, f_m\left(l\left(a^M|s_t\right)\right)\right) \end{aligned}$$

where $\pi\left(a^i|s_t\right)$ is the probability of selecting action i. As a result, the probability of selecting non-critical relay sites is reset to zero.

Bottleneck-Structure-Based Reward Computation. In this chapter, the network saturation throughput is theoretically acquired via the construction of bottleneck structures. Specifically, a bottleneck structure is rebuilt via the algorithm of bottleneck structure construction after taking one step of relay placement a_t to obtain throughput T_{t+1}. The throughput gain $(T_{t+1} - T_t)$ is then taken as the step reward r_t.

Based on the design of the second and third key modules, the contributions of bottleneck theory for the DRL design are summarized as follows. First, the clique gradients are used in the action masking scheme to guide the agent to select the sites in proximity to the most critical bottlenecks. Second, the clique-based bottleneck structure is constructed after each decision step, based on which the reward is computed.

2.3.3 DRL Training and Inference

Offline Training Process

DeepRP is trained through the actor-critic algorithm Proximal Policy Optimization-Clip (PPO-Clip) (Schulman et al., 2017). PPO-Clip has been guaranteed with convergence to the local minimum in recent research (Liu et al., 2019; Huang et al., 2024). Especially, it can attain global optimality with over-parameterized neural networks (Liu et al., 2019). The relay placement process of adding N_r relays is viewed as one DRL episode.

In each step t of an episode, as shown in Fig. 2.3, the DRL agent embeds the input states $\mathbf{s}_t$ through a multi-layer GCN to obtain a graph-level embedding. This embedding goes through the actor and critic network respectively. The actor network outputs the logits for each potential action, and then the clique-gradient-based action masking is applied to obtain a new probability distribution vector over all the actions. Based on the new distribution, action a_t is sampled, i.e., a valid relay site is determined and a new relay is added to the backhaul mesh. Next, a heuristic flow rerouting algorithm named Fair-Share-based (FS-based) rerouting is designed as follows. A new type of link cost c_l for link l is defined as $c_l = \sum_{k \in \hat{\mathcal{Q}}_l} 1/s_k$, where $\hat{\mathcal{Q}}_l$ is the set of cliques containing link l, and s_k is the clique k's fair share with no flow converged. High-impact flows are identified as the flows that traverse the high-impact cliques. Each of these high-impact flows are rerouted by the Dijkstra's algorithm to minimize the total link cost.

After flow rerouting, the clique-based bottleneck structure is constructed to obtain the reward of this step as in Section 2.3.2. When one episode finishes, the episode reward is computed as $r_\mathrm{ep} = \sum_{t=1}^{N_\mathrm{r}} (T_{t+1} - T_t) = T_{N_\mathrm{r}}$, which is the network throughput under the augmented architecture with N_r relays. After collecting a set of episodes, the neural network parameters are updated by stochastic gradient descent. Two loss functions are specified. First, the actor loss is the standard PPO-Clip objective referred to equation (7) in Schulman et al. (2017), where the advantage is estimated by time difference residual, i.e., $A_t^\pi = r_t + V^\pi(\mathbf{s}_{t+1}) - V^\pi(\mathbf{s}_t)$. The gradient with respect to the actor loss is used to update $\boldsymbol{\theta}_\mathrm{g}, \boldsymbol{\theta}_\mathrm{A}$. Second, the critic loss is the mean-square error between the reward-to-go and state values across this set of episodes as in Schulman et al. (2017). The gradient with respect to the critic loss is used to update $\boldsymbol{\theta}_\mathrm{g}, \boldsymbol{\theta}_\mathrm{V}$. The DRL agent is continuously trained over episodes until convergence.

Online Inference Process

For online inference, network designers first need to determine a fixed set of candidate relay sites for the network area of interest. Based on this given information, DeepRP is then run online for N_{r} steps with the trained DRL agent, and the selected relays are physically deployed in the backhaul network.

The online time complexity of DeepRP is analyzed as follows. The complexity of bottleneck structure construction and clique gradient computation is analyzed in Section 2.2.1 and 2.2.2 respectively. For the neural network inference, the time complexity depends on the number of weight multiplications. Through GCNs, the number of multiplications is $O(c_1|\mathcal{V}_{\mathrm{w}}|)$; through the actor network, the number of multiplications is $O(c_2|\mathcal{R}_{\mathrm{can}}|)$, where c_1, c_2 are NN-related constants. In addition, the re-routing algorithm has the same time complexity as Dijkstra's algorithm, $O(L \log N)$, where N, L are the number of deployed nodes and links respectively. As DeepRP needs to be run online for N_{r} times, the overall time complexity is $O(N_{\mathrm{r}}KF\log(KF) + N_{\mathrm{r}}L\log N + c_1N_{\mathrm{r}}|\mathcal{V}_{\mathrm{w}}| + c_2N_{\mathrm{r}}|\mathcal{R}_{\mathrm{can}}|)$.

2.4 Performance Evaluation

2.4.1 Simulation Setup

Network Setup

In this chapter, relay placement is conducted for each backhaul mesh topology within the 2D range 1km×1km. To generate these typologies, a real-world dataset is used in which 46050 BSs are densely deployed in an urban environment (Zhang et al., 2021). 1000 different topologies each of 1km×1km are sampled from this dataset, and they are further divided into the training (70%) and test set (30%). In each sampled network topology, the candidate relay sites are uniformly distributed over the area. For each topology, a THz backhaul mesh network as in Section 2.1 is set up with the bandwidth as 1 GHz and carrier frequency as 0.1 THz. All the network nodes including the relays have the total transmit power of 36 dBm. The noise power spectrum density is -168 dBm/Hz. The SNR threshold for communication links is set to 20 dB. The interference range is characterized by the interfering distance and interfering angle, fixed at 300 m and $\pi/12$ for each side of the main lobe. The end-to-end aggregate flows are generated as follows: there exist one aggregate flow from each mesh node to the mesh gateway, and multiple flows from each mesh node to a few other reachable mesh nodes. For each backhaul topology, 100 different patterns of flow distribution are generated given a certain number of aggregate flows. Before relay placement, the single-path routing is conducted for each flow.

20011372

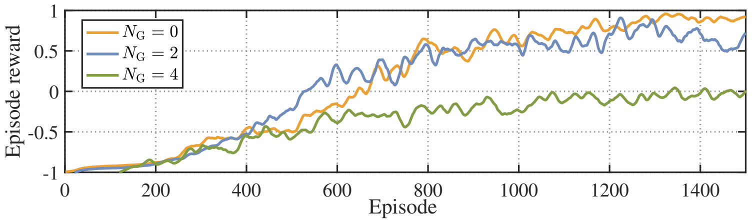
DRL Training Setup

The graph embedding network has three layers with 4, 64, 4 neurons in each layer respectively. With the number of candidate relay sites fixed as 50, the actor network has five layers with 200, 128, 128, 128, 50 neurons in each layer respectively. The critic network has four layers with 200, 256, 256, 1 neurons in each layer respectively. Note that in the last layer of the actor network, Softmax activation is used; except that, in each layer of the above three networks, ReLU activation is used. In summary, the total number of parameters is reported as 183,287. The learning rates for actor and critic networks are set to 0.0001 and 0.003 respectively, and the discount factor is set to 0.99.

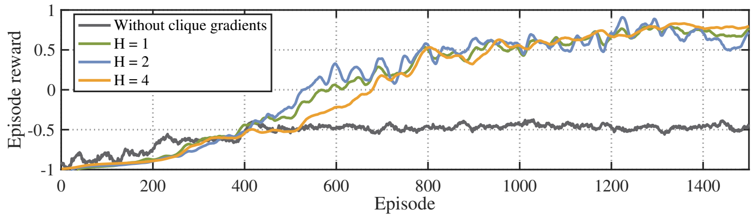
Baseline Approaches

Various baselines of relay placement are explained as follows. First, the random relay placement scheme (*RandomRP*) uniformly samples $N_{\rm r}$ sites from all the candidate sites in the topology to place relays. Second, the heuristic relay placement scheme without clique gradients (*HeurRP-wo-grad*) (Magán-Carrión et al., 2016) is an iterative approach which greedily chooses one relay site with the maximum connectivity degree to the deployed nodes at each iteration, until $N_{\rm r}$ relays are placed. If two relay sites have the same connectivity degree, this scheme selects the relay site with better average link quality. Third, the heuristic relay placement scheme with clique gradients (*HeurRP-with-grad*) follows the similar greedy approach as the second one, but it narrows down the site selection range to the proximity of the H-highest-gradient cliques. Based on relay placement, several flow rerouting schemes are compared as follows. These schemes can be viewed as the variants of Dijkstra's algorithm with different definitions of link costs: 1) in shortest-path rerouting, the link cost is constant as one; 2) in traffic-load-based (TL-based) rerouting (Gao and Zhang, 2006), the link cost is the number of traversed flows on this link; 3) in link-capacity-based (LC-based) rerouting, the link cost is taken as the reciprocal of the link capacity (Toh et al., 2009). For the above schemes, the rerouted paths should not be more than K hops, which is exerted as the maximum-hop constraint for Dijkstra's algorithm (Gao and Zhang, 2006). K is set to the original path length plus two hops in this chapter.

Key simulation parameters are set as follows unless otherwise stated: $f_c = 0.1\text{THz}, G_r = G_t = 18\text{dB}$, absorption coefficient $k_{\rm abs}(f_c) = 0.0033\ \text{m}^{-1}$, time duration $T_s = 50\text{ns}$, shadowing standard deviation $\sigma_X = 7\text{dB}$, cluster arrival rate $\lambda = 0.13\ \text{ns}^{-1}$, cluster decay factor $\Gamma = 3.12\text{ns}$, path loss exponent $n = 2$; the number of relays to be deployed $N_{\rm r} = 10$; the uniform distribution density of candidate relay sites (also termed as candidate site density) is 0.0001 per m^2; the number of end-to-end flows is 200. The results are averaged over the test topologies with the deployed node density around 0.5×10^{-4} per m^2, and each test topology is averaged



(a) Various numbers of GCN layers



(b) Various numbers of high-impact cliques

Figure 2.4: Episode rewards of DRL training.

over 100 uniform relay distribution. Note that the episode rewards are normalized between -1 and 1 during training.

2.4.2 Validation of DeepRP

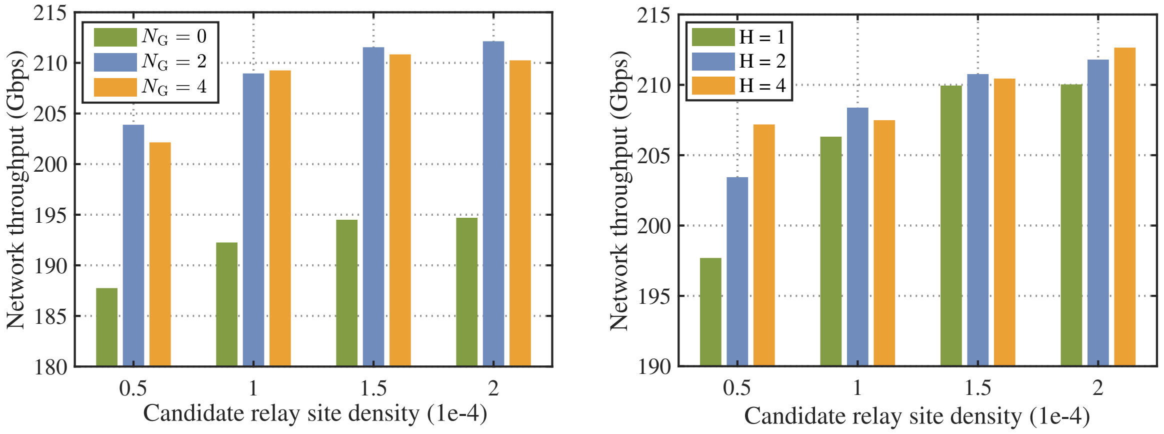
Node Embedding Module

Three different cases are considered, where the numbers of GCN layers are 0, 2, and 4, respectively. The node embedding module is removed in the first case. As shown in Fig. 2.4(a), the training of DRL agent can reach convergence under all three cases, with the episode rewards stable after 1200 training episodes. However, the converged rewards without node embedding is significantly lower than the other two. This difference is also reflected in the network throughput obtained by three schemes, as shown in 2.5(a). Under various relay densities, DeepRP with GCN-based node embedding achieves $7.88-8.29\%$ higher throughput than that without it, which validates the necessity of node embedding. As 2-layer and 4-layer GCNs can achieve similar performance, the 2-layer GCN is selected for DeepRP to reduce computation complexity. This design also matches the physical interference pattern, where the nodes within two hops may interfere with each other even with beamforming.

H-Highest-Gradient Cliques

As described in Section 2.3.3, H-highest-gradient cliques are identified and used in action masking. Different values of H are considered, i.e., $H = 1, 2, 4$. The case without clique gradients is also considered, where an action can be selected out of all the candidate sites. However, as shown in Fig. 2.4(b), the training without clique

20011372



(a) Various numbers of GCN layers (b) Various numbers of high-impact cliques

Figure 2.5: Network throughput achieved by DeepRP.

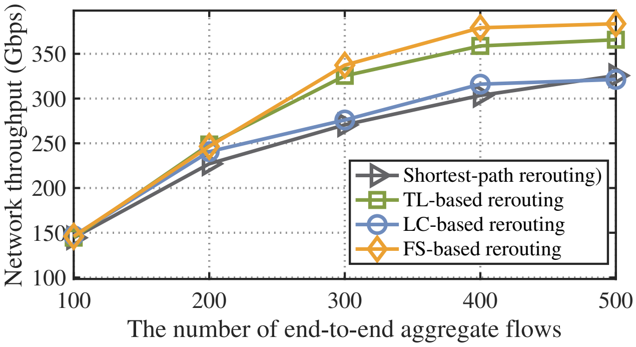


Figure 2.6: Network throughput with different flow rerouting methods.

gradients cannot converge within 1400 episodes, which validates the necessity of embedding clique gradients into DeepRP. With clique gradients, the agent is guided to place relays targeted at the most critical bottlenecks. Thus, more useful actions can be generated during training, resulting in an efficient convergence. The remaining cases are compared in Fig. 2.5(b). As the candidate site density increases, the performance with $H = 1$ increases by 7.14%, while the case with $H = 4$ does not see much gain. This can be interpreted as follows: when the candidate site density is low, there may not be enough relays near the highest-gradient clique, and in this case setting H to a larger value brings the opportunity of exploring more relays; when the candidate site density is high enough, setting a large H cannot bring much gain. Therefore, the value of H should be adjusted with the candidate site density in practice. In the following evaluation, H is set to 2 for the site density of 0.0001 per m^2 or above, and set to 4 otherwise. It should be noted that the trained agent is capable of adapting to new network scenarios without retraining, as the DRL agent is trained with diverse backhaul topologies and flow distributions.

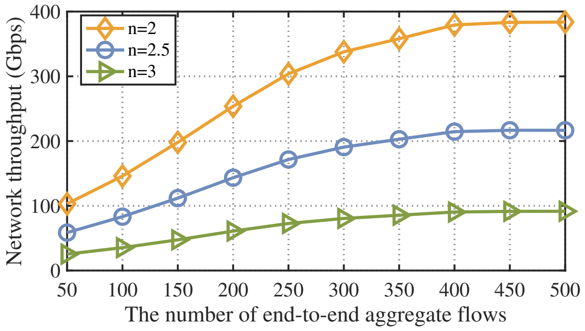


Figure 2.7: Impact of various path loss exponent values.

Flow Rerouting Schemes

Four flow rerouting schemes are compared with various numbers of flows in the network. As the number of flows increases from 100 to 500, RS-based rerouting can achieve the throughput at most 5.53% higher than TL-based rerouting, and 16.82% higher than the other two schemes. Therefore, RS-based rerouting is adopted in the following evaluation.

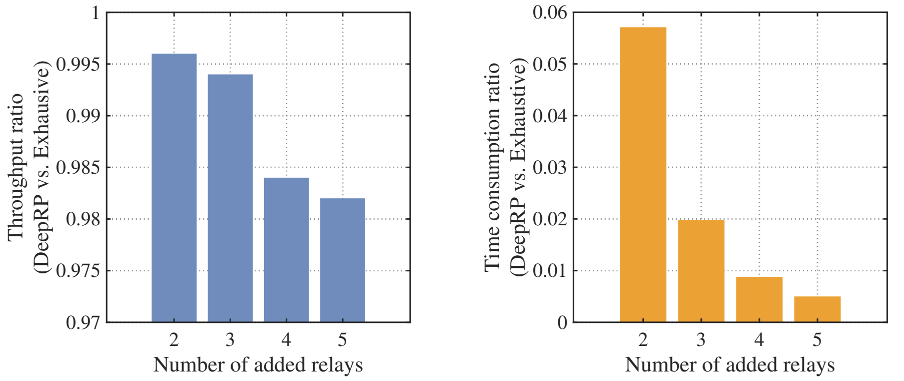
Path Loss Exponent

The impact of various path loss exponent n is studied, specifically $n = 2, 2.5, 3$, where $n = 2$ means the free space propagation and a larger n means a more lossy environment. The added simulations are shown below. The number of added relays is fixed at 30. As the flow density in the network increases, the network throughput reaches a stable level. As shown in Fig. 2.7. The throughputs when $n = 2.5$ and $n = 3$ are approximately 56.4% and 23.8% of that when $n = 2$. The reason for throughput decrease is straightforward: as n increases, the path losses become larger, and thus the receive SNRs and link rates become lower.

2.4.3 Comparisons with Baseline Approaches

Global Optimality for Small-Scale Problems

To empirically evaluate the optimality of DeepRP, we compare it with the exhaustive search scheme for small-scale problems (15 candidate sites in total and the number of added relays varying from 2 to 5). As shown in Fig. 2.8, DeepRP achieves the throughput that is 98.2%-99.6% of the optimal throughput obtained by exhaustive search. The inference time consumption of DeepRP is below 1% of the time consumed by exhaustive search when the number of added relays is 5. For large-scale problems, the time consumption of exhaustive search is not tractable.



(a) Throughput of DeepRP relative to exhaustive search (b) Time consumption of DeepRP relative to exhaustive search

Figure 2.8: Comparisons with the exhaustive search in small-scale problems.

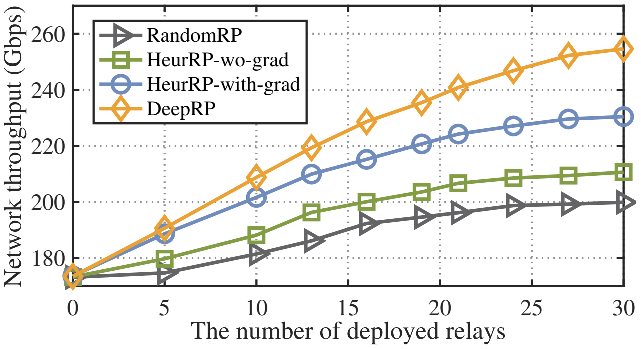


Figure 2.9: Network throughput with various numbers of deployed relays.

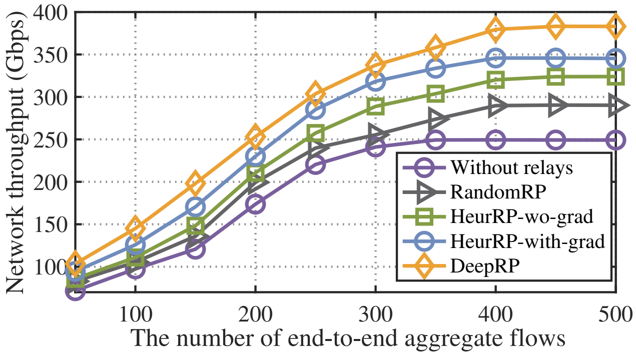


Figure 2.10: Network throughput with various numbers of flows.

Network Throughput

First, the network throughput with different numbers of added relays is compared as shown in Fig. 2.9. As the number of added relays increases from 0 to 20, there exists a rapid increase in network throughput with DeepRP, but the upward trend becomes much slower with the number of added relays above 20. This is due to

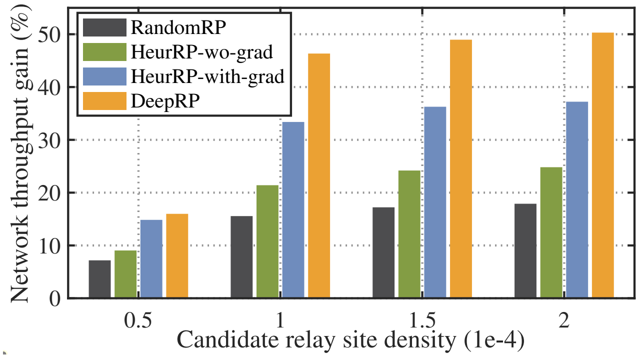


Figure 2.11: Network throughput gain over the original network under various candidate relay site densities.

the fact that the relays at critical locations have been exploited. Furthermore, DeepRP outperforms HeurRP-with-grad by a margin of $5.76 - 10.37\%$ in network throughput when the number of added relay is larger than 10. Interestingly, HeurRP-with-grad can achieve $8.67 - 10.11\%$ higher throughput than HeurRP-wo-grad, which indicates the effectiveness of clique gradients in guiding relay placement. Second, the network throughput with various numbers of flows is evaluated as in Fig. 2.10. In this case, the number of added relays for four schemes is fixed at 30. As the flow density in the network increases, the network throughput reaches a stable level, where the throughput obtained by DeepRP is 10.42% higher than HeurRP-with-grad, and 21.51% and 32.11% higher than HeurRP-wo-grad and RandomRP on average, respectively. Third, the throughput gain over the original network without relays is evaluated under various relay densities, as shown in Fig. 2.11. In each topology, 20 relays are deployed. As the sites become denser especially over 0.0001 per m^2, the throughput gain achieved by DeepRP becomes more significant, which is $40.65 - 50.84\%$. This is because a dense candidate site distribution can increase the chance of finding relays near the most critical bottleneck. Furthermore, when the candidate site density is above 0.0001 per m^2, DeepRP can achieve a 13.36% larger gain than HeurRP-with-grad, and a 25.38% larger gain than HeurRP-wo-grad.

Depth of Bottleneck Structure

As proved in the original theory (Ros-Giralt et al., 2019), the smaller the depth of bottleneck structure is (i.e., the longest length of a directed path), the better the network congestion performance is, which also fits the clique-based case. With 30 deployed relays and 200 flows, the comparison between the average depths of clique-based bottleneck structures is shown in Fig. 2.12. As the site density increases, the gap between the depth of the original network's structure and the depth of augmented network's structure by DeepRP becomes larger. In all the cases, DeepRP achieves the shallowest structure, and it shrinks the structure depth by 41.67% than the original one when the density reaches 0.0002 per m^2.

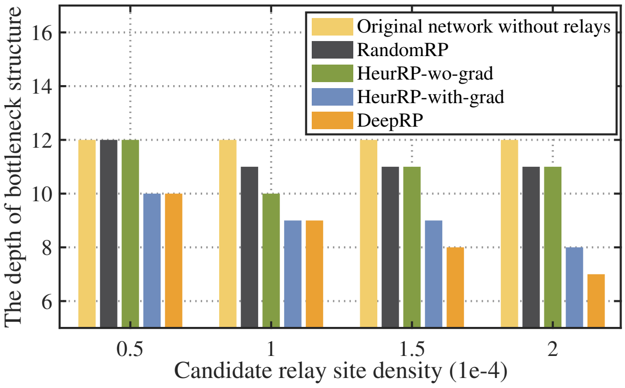
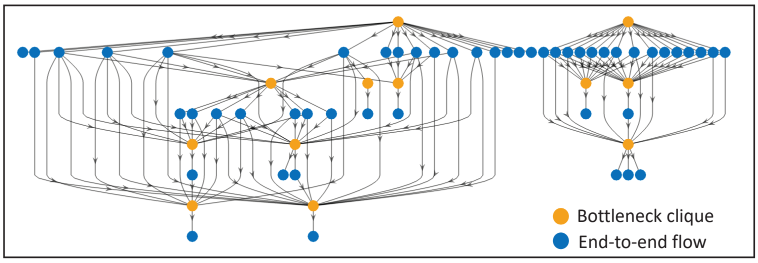
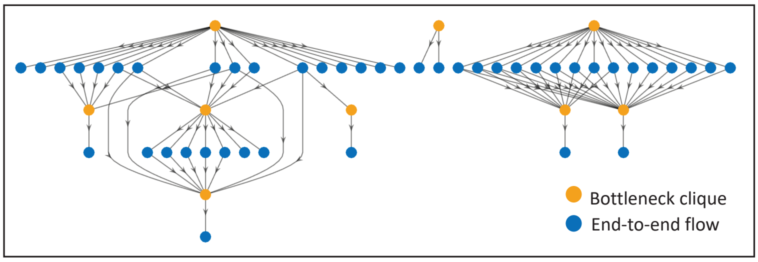


Figure 2.12: Depth of clique-based bottleneck structures.



(a) Bottleneck structure of the original network



(b) Bottleneck structure obtained by DeepRP

Figure 2.13: Comparison of clique-based bottleneck structures.

Clique-Based Bottleneck Structure

The bottleneck structures of the original network and the augmented network by DeepRP are compared in Fig. 2.13. To ease illustration, the number of flows is set to 50, and the number of added relays is 10. The original bottleneck structure has 7 levels with the longest directed path length as 7. After relay placement by DeepRP, the interactions among bottlenecks and flows are much simplified such that a 5-level structure with fewer directed paths. DeepRP can achieve more simplified structures than the baseline methods, but such results are omitted here due to space limit.

20011372

2.5 Summary

The problem of augmenting 6G backhaul architecture with relays to boost network throughput was studied in this chapter. Via developing the clique-based bottleneck theory, the interactions among bottlenecks in the wireless case was captured, from which the relationship between network architecture and network throughput was obtained. The clique gradients were then defined and computed based on the constructed structure, to identify the critical bottlenecks in the network. By integrating the clique-based theory, a DRL-based relay placement approach DeepRP was designed to accomplish relay placement. It is the first work that establishes the bottleneck theory for wireless networks and performs relay placement based on this theory.

20011372

Chapter 3

Network Slice Enforcement for Radio Access Networks Based on Reinforcement Learning

Considering network slicing in a cellular network, one of the most intriguing tasks is slice enforcement over air interfaces across multiple cells. The challenges lie in several aspects. First, resources allocated to different slices must achieve soft isolation at the link level. Second, users' diverse QoS requirements must be satisfied even when communication links experience fading and interference. Third, long-term slicing policies must be conformed, no matter how unbalanced they are. To address these challenges, link-level slice enforcement is first formulated as a resource allocation problem that minimizes radio resource consumption while ensuring link-level soft slice isolation, guaranteeing users' diverse QoS requirements, and conforming to slicing policies. Next, this problem is tackled via a deep reinforcement learning (DRL) based approach, through which LinkSlice is designed as an iterative two-stage algorithm. The first stage determines transmission rates for each link based on DRL. It is embedded with a graph neural network (GNN) to characterize link interference. Based on the transmission rates from the first stage, the second stage allocates resources to each slice. Performance results show that LinkSlice converges quickly to a near-optimal solution. It gracefully tackles the three challenges of link-level slice enforcement while further improving throughput by 18.5%.

3.1 System Model

3.1.1 5G Multi-Cell RAN

A 5G multi-cell RAN is considered in this work. The network includes multiple distributed units (DU) and one central unit (CU, i.e., the central controller of the RAN). All the DUs are connected with the CU via F1 interface (3GPP TR 38.801, 2017). In the rest of the chapter, the DU is referred to as the base station (BS), and the CU is referred to as the central controller. Let $\mathcal{B} = \{1, 2, ...B\}$ denote the set of base stations. The central controller takes centralized scheduling decisions and sends them to the base stations. All the base stations reuse the radio resources in the RAN.

The set of admitted network slices in the RAN is denoted by $\mathcal{M} = \{1, 2, ...M\}$. Let $\mathcal{U}$ be the set of users in the RAN. Each user is assumed to be associated with only one BS. Let $\mathcal{U}_{b,m}$ denote the set of users associated with BS b and slice m. The set of users associated with BS b is denoted as $\mathcal{U}_b = \cup_{m \in \mathcal{M}} \mathcal{U}_{b,m}$. Based on the signal strength, a user is predefined as a cell-edge user or cell-center user (Chang et al., 2009; Yu et al., 2013; Rahman and Yanikomeroglu, 2010). Let $\mathcal{U}_{b,\mathrm{e}}, \mathcal{U}_{b,\mathrm{c}}$ denote the sets of cell-edge users and cell-center users associated with BS b respectively. The sets of all cell-edge and cell-center users are denoted by $\mathcal{U}_\mathrm{e}$ and $\mathcal{U}_\mathrm{c}$ respectively, where $\mathcal{U}_\mathrm{e} = \cup_{b \in \mathcal{B}} \mathcal{U}_{b,\mathrm{e}}$, $\mathcal{U}_\mathrm{c} = \cup_{b \in \mathcal{B}} \mathcal{U}_{b,\mathrm{c}}$.

Time and frequency scheduling is conducted given the predetermined power levels P_u for user u. Cell-center and cell-edge users are allocated with different power levels (Chang et al., 2009). Specifically,

$$P_u = \begin{cases} P_\mathrm{e}, & \forall u \in \mathcal{U}_\mathrm{e}, \\ P_\mathrm{c}, & \forall u \in \mathcal{U}_\mathrm{c}, \end{cases} \tag{3.1}$$

and $P_\mathrm{c} \geq P_\mathrm{e}$. We focus on the downlink transmission that adopts orthogonal frequency division multiple access (OFDMA). Let N_sc denote the number of PRBs within one slot of a radio frame, and T_f denote the number of time slots. Thus, the available resources in one radio frame are divided into N_PRB PRBs with $N_\mathrm{PRB} = N_\mathrm{sc} \cdot T_\mathrm{f}$.

The wireless channel coefficient between the user u and BS b on PRB i is denoted by $h_{u,i}^{(b,m)}$ (the user u is associated with slice m). It is assumed that the channel remains the same within one PRB. Let P_N denote the noise power on each PRB. The actual transmission power of user u is denoted as $\tilde{p}_u$, which depends on PRB scheduling elaborated in Section 3.2.1. Thus, we derive the signal-to-interference-and-noise ratio (SINR) $\gamma_{u,i}^{(b,m)}$ of the user itself on each PRB:

$$\gamma_{u,i}^{(b,m)} = \frac{\tilde{p}_u h_{u,i}^{(b,m)}}{P_{\mathrm{N}} + \sum_{v \neq u, v \in \mathcal{U}} \tilde{p}_v h_{v,i}^{(b,n)}}. \tag{3.2}$$

Let $r_{u,i}^{(b,m)}$ denote the transmission rate for user u on PRB i. The transmission rate is chosen from a set of discrete values according to 3GPP standards (3GPP TR 38.801, 2017). First, the SINR is mapped to the channel quality indicator (CQI) based on the CQI table in 3GPP TR 38.801 (2017). Then, a modulation and coding scheme (MCS) is decided based on CQI. The rate $r_{u,i}^{(b,m)}$ is finally calculated based on the MCS chosen on this PRB. Let $l(\cdot)$ denote SINR to CQI mapping, and $f(\cdot)$ denote CQI to MCS mapping. Both functions are discrete step functions. Therefore, the transmission rate can be expressed as the function of SINR as follows:

$$r_{u,i}^{(b,m)} = f(l(\gamma_{u,i}^{(b,m)})). \tag{3.3}$$

To simplify interference modeling, the interfering links whose interference is below a certain threshold are ignored. Under this assumption, the inter-cell interference experienced by cell-center users is negligible (Yu et al., 2013; Chang et al., 2009). The cell-edge users, however, are prone to interference and performance degradation. Moreover, SINR $\gamma_{u,i}^{(b,m)}$ in equation (3.3) can only be obtained based on the results of PRB scheduling, and it cannot be measured in advance.

In the light of the above facts, nominal SNR is further defined as $\beta_{u,i}^{(b,m)}$ according to (Eslami Rasekh et al., 2020). It is expressed as the ratio of transmit signal strength and noise:

$$\beta_{u,i}^{(b,m)} = \frac{P_u h_{u,i}^{(b,m)}}{P_{\mathrm{N}}}, \forall u \in \mathcal{U}, i = 1, ..., N. \tag{3.4}$$

Thus, transmission rates for cell-center users are directly calculated using nominal SNR (Chang et al., 2009; López-Pérez et al., 2013):

$$r_{u,i}^{(b,m)} = f(l(\beta_{u,i}^{(b,m)})), \forall u \in \mathcal{U}_{b,\mathrm{c}}, i = 1, ..., N. \tag{3.5}$$

For cell-edge users, the SNR degradation caused by interference is expressed as a discrete value $\delta_u^{(b,m)}$, which is the number of levels below the nominal CQI level $l(\beta_{u,i}^{(b,m)})$. Here, we assume that SNR degradation for one user is homogeneous in interfering PRBs (López-Pérez et al., 2013), while nominal SNR for each PRB can still be different and is dependent on time-varying channel conditions. Therefore, transmission rates for cell-edge users are computed as follows (Chang et al., 2009; López-Pérez et al., 2013):

$$r_{u,i}^{(b,m)} = f(l(\beta_{u,i}^{(b,m)}) - \delta_u^{(b,m)}), \forall u \in \mathcal{U}_{b,\mathrm{e}}, i = 1, ..., N. \tag{3.6}$$

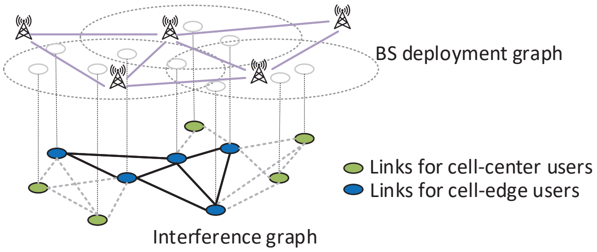


Figure 3.1: BS deployment and interference graph in multi-cell RAN.

Since nominal SNR values can be directly measured by the receiver, the determination of transmission rates for each user is converted to the determination of SNR degradation. Naturally, SNR degradation $\delta_u^{(b,m)}$ reflects the level of aggregated interference from other cells based on PRB scheduling. Furthermore, given the transmission rate $r_{u,i}^{(b,m)}$ assigned to user u, the SINR lower bound $\tilde{\gamma}_{u,i}^{(b,m)}$ can be calculated. Let function $F^{-1}(\cdot)$ denote the reverse mapping process as follows:

$$\tilde{\gamma}_{u,i}^{(b,m)} = F^{-1}(r_{u,i}^{(b,m)}). \tag{3.7}$$

Based on the SINR lower bound, the maximum tolerance interference $I_{u,i}^{(b,m)}$ is derived as follows:

$$I_{u,i}^{(b,m)} = \frac{\tilde{p}_u h_{u,i}^{(b,m)}}{\tilde{\gamma}_{u,i}^{(b,m)}} - P_{\mathrm{N}}. \tag{3.8}$$

3.1.2 Slice Service Model

The QoS requirement of slice m is considered to be the minimum data rate R_m for each radio frame. Let $\mathcal{W}_m = \cup_{b\in\mathcal{B}}\mathcal{U}_{b,m}$ denote the set of users who are associated with slice m across all base stations. The users in $\mathcal{W}_m$ are assumed to be homogeneous in the QoS requirement R_m.

During slice enforcement, the long-term slicing policy $p_{b,m}$ needs to be ensured. The range of $p_{b,m}$ is $[0, 1]$, which prescribes the percentage of resources reserved on BS b for admitted slice m (D'Oro et al., 2019). Each slice is deployed on multiple base stations according to this predetermined slicing policy. The slicing policies are assigned periodically, where the slicing policy period is set as T_{s} radio frames and it can be much larger than the timescale of slice enforcement (i.e., one radio frame). Each slice needs to conform to the slicing policy during each slicing policy period T_{s}, i.e., the resource usage of the slice does not exceed the predetermined percentage of total resources. By conforming to the slicing policy, the traffic load variation of one slice does not affect other slices. Therefore, the slicing policy provides the quality of service on the slice level.

3.1.3 Interference Graph Model

Interference graphs are effective models to capture the interference among links (Chang et al., 2009; Eisen and Ribeiro, 2020; Lee et al., 2021), as shown in Fig. 3.1. Let the undirected interference graph be $\mathcal{G}_{\mathrm{I}} = (\mathcal{V}_{\mathrm{I}}, \mathcal{E}_{\mathrm{I}})$, where $\mathcal{V}_{\mathrm{I}}$ denotes the set of nodes, and $\mathcal{E}_{\mathrm{I}}$ denotes the set of edges. The node V_u in $\mathcal{V}_{\mathrm{I}}$ represents the downlink communication link between user u and its associated BS. The edge between nodes V_u and V_v represents the existence of interference between these two communication links. The estimated interference strength between two communication links is represented as a weight $w_{u,v}$ on its corresponding edge. There is no edge among the transmission links within the same cell, as no intra-cell interference is allowed. Let $\mathcal{V}_{\mathrm{I}}(u)$ denote the neighboring (i.e., interfering) links of user u's link on the interference graph. Thus, the SINR is reformulated as follows:

$$\gamma_{u,i}^{(b,m)} = \frac{\tilde{p}_u h_{u,i}^{(b,m)}}{P_{\mathrm{N}} + \sum_{v \in \mathcal{V}_{\mathrm{I}}(u)} \tilde{p}_v h_{v,i}^{(b,n)}}, \tag{3.9}$$

where $\tilde{p}_v h_{v,i}^{(b,n)}$ is related to $w_{u,v}$ based on PRB scheduling. Note that equation (3.9) is evolved from equation (3.2). Equation (3.2) is the general SINR expression that considers the potential interference from all the other users in the network, while equation (3.9) only retains the potential interference from neighboring nodes on the interference graph.

3.2 Problem Formulation and Transformation

3.2.1 Problem Formulation

The fine-grained, link-level slice enforcement problem is studied in a multi-cell network. Slice enforcement is conducted for each radio frame t, which involves multiple types of entities, namely base stations, slices, and users. Given the predetermined long-term slicing policy $p_{b,m}$, PRBs on the involved base stations are allocated to each link. For the benefit of the physical network operator, once link-level QoS requirements are guaranteed, precious radio resources need to be saved as much as possible.

Furthermore, the requirement on slice isolation is stated as follows: interfering links associated with different slices are not allowed to use the same PRB resources for safety and privacy considerations (D'Oro et al., 2018). Since soft slice isolation is conducted across multiple cells, the impact of inter-cell interference on the performance of isolation needs to be considered. For two different slices, if they are allocated with the same set of PRB resources across cells, there can be inter-slice interference caused by inter-cell interference. For a single slice, if it is allocated with the same set of PRB resources across cells, there can also be intra-slice interference caused by inter-cell interference. Different approaches are taken to tackle inter-slice interference and intra-slice interference. Inter-slice interference is avoided by link-level resource separation, which is indispensable due to the slice isolation requirement. However, there is no isolation issue on the physical layer for the same slice. If there exists intra-slice interference, one approach is allocating time/frequency resources to the interfering links to guarantee the user demands, and the other one is performing power allocation among intra-slice links to mitigate this interference. This chapter is focused on the former approach. Based on the above analysis, the objective of slice enforcement is to minimize the long-term PRB usage over each slicing policy period T_s, i.e., the number of consumed PRBs, for the whole RAN. In the meantime, the long-term slicing policy, the QoS requirement, and slice isolation have to be guaranteed. For the conformance of the slicing policy, the average PRB usage of each slice is smaller than the threshold prescribed by the slicing policy $p_{b,m}$ over each slicing policy period T_s. For the QoS requirement, in each radio frame t, a user subscribing to the service of slice m has the minimum data rate requirement denoted by R_m. Thus, the resources allocated to user u need to transmit at least $R_m\sigma$ bits in one radio frame, where σ denotes the time duration of one radio frame. Also, the intra-slice interference must be smaller than the maximum tolerance interference $I_{u,i}^{(b,m)}$ in equation (3.8). For slice isolation, the interfering links associated with different slices are not allowed to use the same PRB resources. Moreover, to avoid intra-cell interference, the transmission links in the same cell must not reuse PRBs. Let $x_{u,i}^{(b,m)}(t)$ denote the PRB allocation indicator for user u associated with BS b and slice m on PRB i, during the radio frame indexed by t. $x_{u,i}^{(b,m)}(t)$ equals 1 when PRB i is allocated to user u, and 0 otherwise. Based on the PRB allocation indicator, the actual transmit power can be expressed by $\tilde{p}_u = x_{u,i}^{(b,m)}P_u$, and the actual interfering power based on the weight $w_{u,v}$ on the interference graph can be denoted as $\tilde{p}_v h_{v,i}^{(b,n)} = x_{v,i}^{(d,n)} w_{u,v}$ (the user v is associated with BS d and slice n). To simplify the notation, the radio frame index t is omitted for the following formulation except for explicit explanation. The slice enforcement problem is formulated as **Problem 1 (P1)**:

$$\min_{x_{u,i}^{(b,m)}} \quad \frac{1}{T}\sum_{t=1}^{T}\left[\sum_{u\in\mathcal{U}}\sum_{i=1}^{N} x_{u,i}^{(b,m)}(t)\right], \tag{3.10}$$

20011372

s.t.:

$$\sum_{i=1}^{N} x_{u,i}^{(b,m)} r_{u,i}^{(b,m)} \geq R_m \sigma, \forall u \in \mathcal{U}_{b,m}, b \in \mathcal{B}, m \in \mathcal{M}, \tag{3.11}$$

$$\sum_{u \in \mathcal{U}_b} x_{u,i}^{(b,m)} \leq 1, \forall b \in \mathcal{B}, i = 1, ..., N, \tag{3.12}$$

$$x_{u,i}^{(b,m)} + \sum_{\substack{v \in \mathcal{V}_{\mathrm{I}}(u), \\ n \neq m, d \neq b}} x_{v,i}^{(d,n)} \leq 1, \forall u \in \mathcal{U}, i = 1, ..., N, \tag{3.13}$$

$$x_{u,i}^{(b,m)} \sum_{v \in \mathcal{V}_{\mathrm{I}}(u), d \neq b} w_{u,v} x_{v,i}^{(d,m)} \leq I_{u,i}^{(b,m)}, \forall u \in \mathcal{U}, i = 1, ..., N, \tag{3.14}$$

$$\frac{1}{T} \sum_{t=1}^{T} \left[\sum_{u \in \mathcal{U}_{b,m}} \sum_{i=1}^{N} x_{u,i}^{(b,m)}(t) \right] \leq N p_{b,m}, \forall b \in \mathcal{B}, m \in \mathcal{M}, \tag{3.15}$$

$$x_{u,i}^{(b,m)} \in \{0, 1\}, i = 1, ..., N, \tag{3.16}$$

where objective (3.10) optimizes the network performance measured over each slicing policy period T_{s}; constraint (3.11) describes the QoS requirement; constraint (3.12) guarantees that there is no intra-cell interference; constraint (3.13) ensures the link-level soft slice isolation; constraint (3.14) guarantees that the intra-slice interference is below the maximum tolerance interference; constraint (3.15) guarantees the long-term enforcement of slicing policy over each slicing policy period T_{s}.

3.2.2 Problem Transformation

Slice enforcement is a decision-making problem over multiple time steps, i.e., radio frames, to maximize a long-term performance metric. This problem is extremely hard to solve considering the following perspectives. First, the objective function aims at minimizing the resource usage spanning multiple time steps, while resources are allocated per time step. Second, there is a mixture of long-term constraints and short-term constraints. Third, there is a coupling effect among the long-term objective, the long-term constraints, and the short-term constraints. Last but not least, the computation complexity of even a sub-problem is NP-hard as we will demonstrate in Section 3.3.

Deep reinforcement learning (DRL) is a perfect candidate to handle such a long-term optimization problem, where the agent in DRL aims to maximize the long-term rewards received from the environment through multi-step actions. Thus, DRL can tackle the coupling between the long-term objective and short-term decision in slice enforcement problems. Furthermore, maximizing the long-term network performance requires future information on time-varying channels and interference conditions, which needs complicated mathematical modeling. The model-free DRL algorithm does not require any prior information on the environment. The DRL agent adapts to the dynamic environment and learns the optimal policy by interacting with the environment. Therefore, mathematical modeling and prediction are avoided.

However, when DRL is applied to solve problem P1, one of the obstacles is that there is a long-term constraint (3.15) on the slicing policy conformance. In this case, it is difficult to decompose problem P1 into multiple separate time steps. To tackle this issue, we define the penalty term

$$L_{b,m}(t) = \max\{0, \sum_{u\in\mathcal{U}_{b,m}} \sum_{i=1}^{N} x_{u,i}^{(b,m)}(t) \ - Np_{b,m}\}, \tag{3.17}$$

for frame t to transform the constraint (3.15) into the objective function. If there is no violation of the slicing policy, the penalty term $L_{b,m}$ equals zero. If slicing policy violation takes place, the corresponding penalty is exerted on the objective function. Let λ denote the penalty coefficient associated with this penalty term. We thus present an approximate problem P2 to problem P1:

Problem 2 (P2):

$$\min_{x_{u,t}^{(b,m)}} \quad \frac{1}{T}\sum_{t=1}^{T}\left[\sum_{u\in\mathcal{U}}\sum_{i=1}^{N} x_{u,i}^{(b,m)}(t) + \lambda \sum_{b\in\mathcal{B}}\sum_{m\in\mathcal{M}} L_{b,m}(t)\right], \tag{3.18}$$

s.t.:

$$\text{Eqs.}(3.11) - (3.14), (3.16).$$

Although problem P2 is an approximated version of P1, it is obvious that the optimal solution to P1 is also the optimal solution to P2. We can also prove that problem P2 can be guided by DRL reward design to approach the optimal solution to P1. With a large enough penalty coefficient λ, the DRL agent can always receive larger rewards when the actions generated by DRL fall into the feasible region of P1 (the reward design is elaborated in Section 3.4.1). This is shown as follows.

Theorem 3.1. *When the local optimal solution to problem P2 falls out of the feasible region $\mathcal{F}$ of problem P1, there exists $\lambda \in \mathbb{R}$ such that, the optimal objective value of P2 must be larger than that of P1.*

The detailed proof is given in Appendix B.

3.3 LinkSlice Overview

As discussed in Section 3.2, DRL can be utilized to handle multi-step optimization problems. However, if DRL is directly applied to solve problem P2 by generating the PRB allocation indicators as the actions, it is difficult for the DRL agent to learn the desired policy in practice. The reasons lie in two aspects: 1) since there exist a large number of constraints in problem P2, guiding the DRL agent towards feasible actions through reward engineering is notoriously difficult (Paternain et al., 2019); 2) even though DRL can reach feasible actions, converging to a near-optimal solution is still challenging, as the combinatorial nature of P2 leads to a large discrete action space (Bello et al., 2017). Therefore, the direct application of DRL is not practical.

To this end, we design a two-stage framework: DRL is applied to obtain the transmission rates in the first stage, and slice enforcement is conducted in the second stage given the transmission rates. Especially, for the second stage, the remaining PRB scheduling problem for each radio frame is still computationally hard. The following sub-problem needs to be solved:

Problem 3 (P3):

$$\min_{x_{u,i}^{(b,m)}} \quad \sum_{u\in\mathcal{U}}\sum_{i=1}^{N} x_{u,i}^{(b,m)} + \lambda \sum_{b\in\mathcal{B}}\sum_{m\in\mathcal{M}} L_{b,m}, \tag{3.19}$$

s.t.:

$$\text{Eqs.}(3.11) - (3.14), (3.16).$$

Problem P3 is an integer quadratic-constrained programming problem (IQCP). The variable space of P3 grows exponentially with the numbers of users and of PRBs. A brief proof of NP-hard property of P3 is given by theorem 3.2. It is computationally expensive to obtain the optimal solution as the size of the network increases. Therefore, a low-complexity heuristic algorithm named greedy slice enforcement with masking (G-SEM) is proposed to solve problem P3.

Theorem 3.2. *Problem P3 is NP-hard.*

The detailed proof is given in Appendix B.

The two-stage framework of *LinkSlice* is shown in Fig. 3.2. The interference graphs, channel conditions, and user demands are collected for the two stages. In the first stage, the DRL outputs the SNR degradation, which is then transformed into transmission rates through the mapping function in Equation (3.6). In the second stage, given the transmission rates for each link in each radio frame, a low-complexity greedy algorithm G-SEM

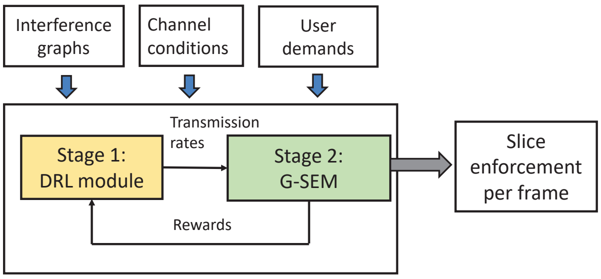


Figure 3.2: LinkSlice overview: a DRL-based heuristic solution for slice enforcement.

is designed to obtain the near-optimal solution to problem P3. G-SEM conducts slice enforcement per radio frame. The rewards are collected from the second stage and fed back into the first stage. The two stages work together iteratively until reaching convergence.

3.4 Two-Stage Slice Enforcement: A DRL-Based Heuristic Solution

Compared with the standard DRL architecture, LinkSlice is unique in two aspects. First, the actor and critic networks contain embedded GCNs to capture the interference among communication links. Then, a pooling mechanism is leveraged to reduce the size of the action space. Second, the optimization algorithm G-SEM is embedded into the DRL environment to conduct PRB scheduling. The detailed blocks are shown in Fig. 3.3. The design of LinkSlice is elaborated in this section.

3.4.1 Stage I: A DRL Approach for Interference-Aware Rate Determination

A DRL problem is generally based on a Markov decision process (MDP) consisting of the state space, action space, reward space, transition probability set, and initial state distribution, i.e., $\{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, p_0\}$. The overall architecture and detailed design of DRL used in LinkSlice are presented in the following.

Overall Architecture of DRL. As illustrated in Fig. 3.3, an actor-critic architecture is utilized. Considering each radio frame as a time step in DRL, the global states $\mathbf{s}(t)$, actions $\mathbf{a}(t)$ and instantaneous rewards $\tilde{r}(t)$ are modeled for the task of SNR degradation estimation. The central controller in the RAN is viewed as the global agent. At the start of each time step, the DRL agent (i.e., the actor and critic) observes the states $\mathbf{s}(t)$ for each node (i.e., one communication link between a user and its associated BS) on the interference graph. The states are then embedded by a two-layer GCN and sum-pooled. The pooled embeddings are fed into a two-layer fully-connected neural network (FCNN), which outputs the probability distribution of actions after SoftMax activation. The actor then samples the actions according to the probability distribution $\pi(\mathbf{a}|\mathbf{s})$ (i.e., the policy). After interacting with the environment, the agent collects rewards to update both the actor and critic networks.

In the following part, we present the design of the state, action, reward, and training algorithm of DRL.

State. The node state $s_u(t)$ in time step t consists of three parts, including the channel condition, the link data rate in the last time step, and the gap between this link data rate and QoS requirement. For clarity, The channel condition is denoted by a vector of nominal SNR $\boldsymbol{\beta}_u(t)$. Each element of $\boldsymbol{\beta}_u(t)$ is a nominal SNR measured on one PRB's bandwidth. The link data rate during the last frame $\tilde{R}_u(t-1)$ can be computed by adding the transmission bits on each PRB together. The gap between the link's QoS requirement and the actual data rate describes the level of QoS satisfaction, which is denoted as $\Delta_{\mathrm{R},u}(t-1) = \tilde{R}_u(t-1) - R_m$ (the user u is associated with slice m). Based on these three parts, the node state is formulated as

$$\mathbf{s}_u(t) = \left[\boldsymbol{\beta}_u(t), \tilde{R}_u(t-1), \Delta_{\mathrm{R},u}(t-1)\right]^{\mathsf{T}}, \tag{3.20}$$

where $\tilde{R}_u(t-1)$ is formulated as

$$\tilde{R}_u(t-1) = \sum_{i=1}^{N} x_{u,i}^{(b,m)}(t-1) r_{u,i}^{(b,m)}(t-1). \tag{3.21}$$

The state $\mathbf{s}(t)$ observed by the agent consists of all the node states, which is presented as

$$\mathbf{s}(t) = \left[\mathbf{s}_1(t), \mathbf{s}_2(t), ..., \mathbf{s}_U(t)\right]^{\mathsf{T}}. \tag{3.22}$$

State Aggregation and Pooling. Given all the node states in $\mathbf{s}(t)$, one straightforward way is to concatenate them to generate a long vector that is used as the input of the DRL networks. However, there are two obvious drawbacks to this design. First, it is not scalable with increasing users. When the number of users increases, the dimension of network inputs also increases. Second, this design ignores the relationships among the interfering links encoded in the interference graph.

Graph neural networks (GNN) (Gao et al., 2018; Liu et al., 2010; Kipf and Welling, 2016; Jiang et al., 2020) are proposed to provide representations for the graph data structure. The GNN embeds the node feature and its structural information into the embedding space. In LinkSlice, an interference-aware DRL network is designed, where GCNs are utilized to capture the relationships among nodes and embed the interference graphs into the states of DRL. Let $\boldsymbol{\theta}_{\mathrm{g}}^{(\ell)}$ denote the parameters of ℓ-th layer GCN. Let d_s denote the dimension of the original node state, and d_l denote the dimension of the embedded node state. The aggregation weights $w_{u,v}$ and the

20011372

neighboring set $\mathcal{V}_\mathrm{I}(u)$ of each node u are provided by the given interference graph. Let g_u denote the degree of user u in the interference graph. Two GCN layers are stacked for generating state embeddings $\tilde{\mathbf{s}}_u(t)$ for each node u, which captures the states of one-hop and two-hop neighboring nodes. The weighted aggregation process in the two GCN layers is presented as follows:

$$\tilde{\mathbf{s}}_u(t) = \sum_{v \in \mathcal{V}_\mathrm{I}(u) \cup \{u\}} \frac{w_{u,v}}{\sqrt{g_u g_v}} \sigma_\mathrm{R} \left[\left(\sum_{v \in \mathcal{V}_\mathrm{I}(u) \cup \{u\}} \frac{w_{u,v}}{\sqrt{g_u g_v}} \sigma_\mathrm{R}(\mathbf{s}_u(t)^\intercal \boldsymbol{\theta}_\mathrm{g}^{(1)}) \right) \boldsymbol{\theta}_\mathrm{g}^{(2)} \right], \tag{3.23}$$

where $\sigma_\mathrm{R}(\cdot)$ is the ReLU activation function. During the state aggregation stage, the agent collects the states of all the nodes on the interference graph and conducts state aggregation via GCNs. The aggregated embeddings are represented by a matrix $\tilde{\boldsymbol{S}}$ (with time step t omitted for simplicity of notation) with the dimension $U \times d_l$. Each row of $\tilde{\boldsymbol{S}}$ corresponds to one node's embedding.

A graph pooling mechanism is added after state aggregation. Each cell is uniformly divided into N_s sectors. For example, the cell can be sectorized into six sectors, with $\pi/3$ radians each. The total number of sectors is denoted as K_s with $K_\mathrm{s} = N_s B$ (B denotes the number of cells). In sector k, the aggregated embeddings are further sum pooled to generate one representation for sector k. The pooled embeddings are denoted by $\tilde{\boldsymbol{S}}_\mathrm{P}$ with the dimension $K \times d_l$ (with time step t omitted). Each row of $\tilde{\boldsymbol{S}}_\mathrm{P}$ corresponds to one sector's embedding.

The "sector" in our design is a virtual concept for graph pooling, instead of "cell sector" from directional antennas. The sector determines the range of sum pooling. It is practical to sum pool the node embeddings in the same sector, as the cell-edge users in the same sector have similar interference topologies that lead to similar SNR degradation. Although the sectorization scheme causes inaccuracy in the interference modeling that will impact the network performance, it can shrink the size of action space significantly. Instead of outputting actions for each node, the actor network can output actions for each sector, as elaborated in the following design of action space. The impact of the sectorization scheme on LinkSlice is evaluated in Section 3.5.4.

Action. The pooled embeddings are then fed into a two-layer FCNN. Let d_a denote the dimension of output layer of FCNNs, i.e., the number of actions that can be chosen from. The two-layer FCNN parameters are represented by two matrices $\boldsymbol{W}_\mathrm{F}^{(1)}$, $\boldsymbol{W}_\mathrm{F}^{(2)}$ with the dimensions $d_l \times d_\mathrm{F}$, $d_\mathrm{F} \times d_a$ (let d_F denote the dimension of the hidden layer). The action probability distribution output by the FCNNs is denoted as $\sigma_\mathrm{S}\left[\sigma_\mathrm{R}(\tilde{\boldsymbol{S}}_\mathrm{P} \boldsymbol{W}_\mathrm{F}^{(1)}) \boldsymbol{W}_\mathrm{F}^{(2)}\right]$, with the dimension $K \times d_a$ (let $\sigma_\mathrm{R}(\cdot)$ denote the ReLU activation function, and $\sigma_\mathrm{S}(\cdot)$ denote the SoftMax activation function). This process should be noted with two features. First, the action is conducted for each sector instead of each node, and all the nodes in the same sector share the selected action. Second, the embeddings for each sector are stacked into $\tilde{\boldsymbol{S}}_\mathrm{P}$, and FCNN parameters are shared among all the sectors, as

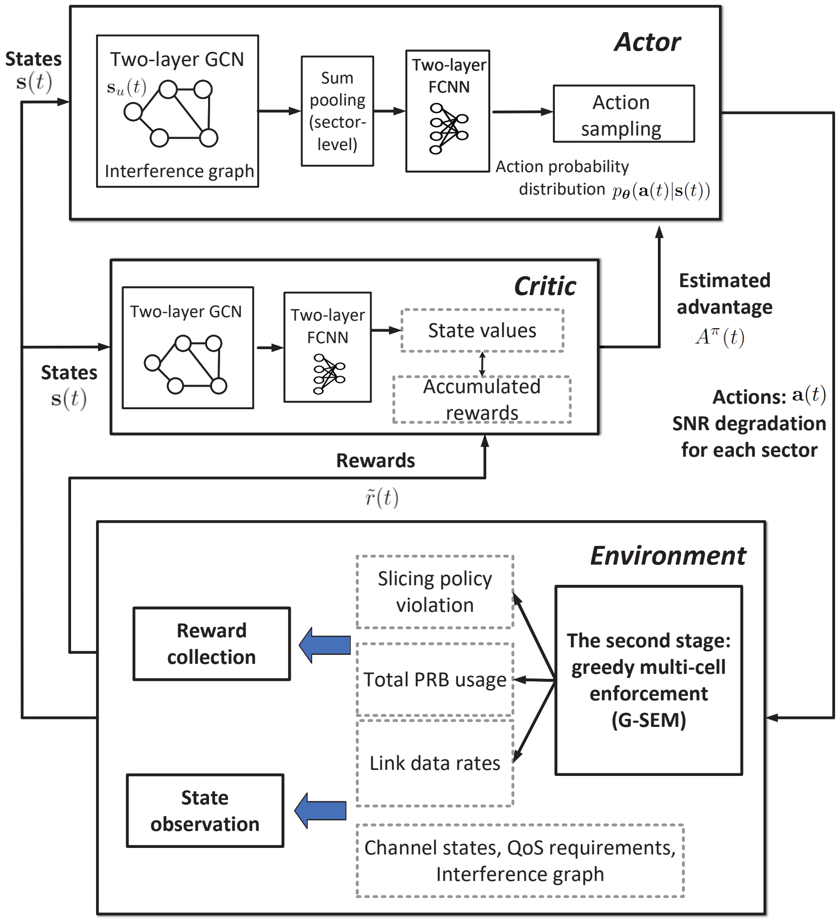


Figure 3.3: LinkSlice detailed design: two-stage slice enforcement.

revealed in the matrix multiplication in $\sigma_{\mathrm{S}}\left[\sigma_{\mathrm{R}}(\tilde{\boldsymbol{S}}_{\mathrm{P}}\boldsymbol{W}_{\mathrm{F}}^{(1)})\boldsymbol{W}_{\mathrm{F}}^{(2)}\right]$ where $\boldsymbol{W}_{\mathrm{F}}^{(1)}$, $\boldsymbol{W}_{\mathrm{F}}^{(2)}$ are operated on each row of $\tilde{\boldsymbol{S}}_{\mathrm{P}}$. This design is similar to graph convolutional reinforcement learning proposed in Jiang et al. (2020).

Since the graph pooling based on the sectorization scheme is leveraged, the actions are sampled for each sector instead of each node. Therefore, the joint action space is shrunk from $(d_a)^U$ to $(d_a)^K$, with K_{s} equal to the number of sectors. The action taken by the agent $\mathbf{a}(t)$ consists of actions sampled for each sector. This is presented as

$$\mathbf{a}(t) = \left[a_1(t), a_2(t), ..., a_K(t)\right], \tag{3.24}$$

where $a_k(t)$ is defined as the SNR degradation for sector k sampled from d_a possible discrete values. Originally, the SNR degradation level $\delta_u^{(b,m)}$ is defined on the link level, which describes the channel degradation for each link between user u and its associated BS b. After the pooling mechanism is introduced, the SNR degradation

20011372

level selected for each sector describes the average channel degradation for all the cell-edge links in this sector. Given the SNR degradation level, the transmission rate is determined based on equation (3.6).

Reward. The long-term objective in problem P2 can be naturally mapped to the accumulated reward in DRL design. However, during the learning stage, it is possible that the second stage is not feasible (i.e., the users' QoS requirements cannot be satisfied) given the transmission rates from the first stage, as shown in Section 3.4.2. To solve this problem, when the DRL agent falls in the infeasible status, an extremely large negative reward is imposed on the agent. Thus, the infeasible status can be efficiently avoided during DRL training.

The instantaneous reward $\tilde{r}(t)$ for each time step t consists of two parts: the objective value in one step of problem P2 (including network-wide PRB usage and slicing policy violation), and the negative reward for infeasible status. Therefore, we have

$$\tilde{r}(t) = -\mathbb{1}\{\mathcal{H} \neq \emptyset\} W(t) - \mathbb{1}\{\mathcal{H} = \emptyset\} \mu, \tag{3.25}$$

$$\begin{aligned} W(t) &= \sum_{u\in\mathcal{U}} \sum_{i=1}^{N} x_{u,i}^{(b,m)}(t) + \lambda \sum_{b\in\mathcal{B}} \sum_{m\in\mathcal{M}} L_{b,m}(t), &(3.26) \\ &= \sum_{b\in\mathcal{B}} \left[\sum_{u\ \in\mathcal{U}_b} \sum_{i=1}^{N} x_{u,i}^{(b,m)}(t) \ + \ \lambda \sum_{m\in\mathcal{M}} L_{b,m}(t) \right], &(3.27) \end{aligned}$$

where $\mathcal{H}$ denotes the feasible set of the second stage (i.e., problem P3), $-\mathbb{1}\{\mathcal{H} = \emptyset\}\mu$ denotes the penalty for the infeasibility of the second stage (the indicator function $\mathbb{1}$ of an event takes value 1 when the event is true and value 0 otherwise). In practice, μ is set as the maximum value of $W(t)$ collected in the environment.

Training Algorithm. The DRL agent with stochastic policies is considered: a policy $\pi(\mathbf{a}|\mathbf{s}) : \mathcal{S} \rightarrow P(\mathcal{A})$ represents the mapping from state space $\mathcal{S}$ to probability distributions over action space $\mathcal{A}$. The policy is parameterized by the actor and critic networks, denoted as $\boldsymbol{\theta}_{\mathrm{A}}$ and $\boldsymbol{\theta}_{\mathrm{V}}$ respectively. Both the actor and critic networks contain two-layer GCNs for node embedding, and two layer FCNNs. Note that the output dimension of the critic network is one. The actor and critic networks are trained separately. The action taken by the agent leads to an instantaneous reward $\tilde{r}(t) \in \mathbb{R}$.

The learning process of DRL is based on the policy gradient algorithm. The goal is to maximize the expected long-term reward $\mathbb{E}\left[\sum_{t=1}^{T} \tilde{r}(t)\right]$ over one episode (i.e., T_{s} timesteps). Generalized advantage function (GAE)

20011372

(Schulman et al., 2015) is estimated as

$$A_t^\pi := \tilde{r}(\mathbf{s}(t), \mathbf{a}(t)) + V^\pi(\mathbf{s}(t+1)) - V^\pi(\mathbf{s}(t)). \tag{3.28}$$

The advantage function $A^\pi(t)$ under policy π reveals how much the current action $\mathbf{a}(t)$ is better than average given the current state $\mathbf{s}(t)$. The value function $V^\pi(t)$ is the discounted reward-to-go approximated by the critic network as follows:

$$V^\pi(\mathbf{s}(t)) = \mathbb{E}\left[\sum_{t=t'}^{T} \gamma^{t'-t} \tilde{r}(\mathbf{s}(t'), \mathbf{a}(t'))\right], \tag{3.29}$$

where γ is the discounted factor.

Once the advantage is estimated, the policy gradient algorithm is applied to update the actor network. The objective of optimizing DRL's policy is:

$$\max_{\boldsymbol{\theta}} \mathbb{E}\left[\sum_{t=1}^{T} p_\theta(\mathbf{a}(t)|\mathbf{s}(t)) A_t^\pi\right], \tag{3.30}$$

where $p_{\boldsymbol{\theta}}(\mathbf{a}(t)|\mathbf{s}(t))$ is the probability of taking action $\mathbf{a}(t)$ based on the policy $\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})$.

The gradient of the objective function is taken as follows:

$$\nabla \mathbb{E}\left[\sum_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{a}(t)|\mathbf{s}(t)) A_t^\pi\right] = \mathbb{E}\left[\sum_{t=1}^{T} A_t^\pi \nabla \log(p_{\boldsymbol{\theta}}(\mathbf{a}(t)|\mathbf{s}(t)))\right]. \tag{3.31}$$

In the first stage, the DRL module outputs the SNR degradation as actions, which are then converted to transmission rates via equation (3.6). Given the transmission rates generated by the DRL module, a greedy priority-based algorithm is designed to conduct slice enforcement per radio frame, which is elaborated in the following part.

Theorem 3.3. *The complexity of greedy multi-cell enforcement algorithm is* $O(B \log B + N^2 + U \log \frac{U}{B} + UN \log N + BN)$*, where* B*,* U*, and* N *represent the number of BSs, users, and PRBs, respectively.*

The proof is given in Appendix B.

3.4.2 Stage II: Slice Enforcement Through Link-Level Scheduling

When slice enforcement is conducted over multi-cell air interfaces, two base stations whose coverage regions do not overlap with each other can enforce slices independently. The BS coverage, i.e., BS topology, is known in advance (Zhou et al., 2016; Chang et al., 2009).

Algorithm 2 G-SEM: Greedy slice enforcement with masking mechanism.

```
Input: The estimated transmission rates $r_{u,i}^{(b,m)}$ output by DRL, the interference graph $\mathcal{G}_{\mathrm{I}}$, long-term slicing policies $\mathcal{P}$, channel conditions of links $h_{u,i}^{(b,m)}$, the BS topology.
Initialization:
Identify the sets of base stations $\{\mathcal{B}_1, ..., \mathcal{B}_C\}$ that operate in parallel by applying Welsh-Powell algorithm on the BS topology. Let $\{1, ..., C\}$ denote the index of these sets.
for $c = 1$ to $C$ do
    Conduct slice enforcement in parallel for base stations in the set $\mathcal{B}_c$;
    for $b \in \mathcal{B}_c$ do
        for $u \in \mathcal{U}_{b,\mathrm{edge}}$ do
            Identify free PRBs for each user via the masking mechanism.
        end for
    end for
    for $b \in \mathcal{B}_c$ do
        Generate the scheduling priority for each user in the set $\mathcal{U}_b$;
        for $u \in \mathcal{U}_b$ do
            Sort the free PRBs based on transmission rates $r_{u,i}^{(b,m)}$ in the descending order;
            Allocate free PRBs according to the above order until each user's QoS requirement is satisfied.
        end for
    end for
end for
if this is the training stage of LinkSlice then
    Collect the results of slice enforcement and generate the instantaneous reward $\tilde{r}$.
end if
Output: A PRB allocation scheme $\mathcal{X}$ for slice enforcement.
```

A greedy slice enforcement algorithm with masking mechanism (G-SEM) is designed with three main blocks. The first block of G-SEM identifies the clusters of BSs that can enforce slices in parallel, which is achieved by Welsh–Powell algorithm (Welsh and Powell, 1967). Welsh–Powell algorithm is a greedy graph coloring algorithm based on the degrees of nodes.

For the BSs assigned with the same color, slice enforcement can be conducted in parallel. In the second block, a priority-based algorithm is designed for slice enforcement in a single cell. The allocation priority of users is determined by the following rule. First, the cell-edge users have a higher priority than the cell-center users. Then, for both the cell-edge and cell-center users, those with higher QoS requirements are assigned higher priorities. Last, for the users with the same QoS requirements, i.e., associated with the same slice, those with larger degrees on the interference graph are assigned higher priorities. Once the user priorities are sorted, slice enforcement is carried out sequentially. For each user, the PRBs that can be allocated are sorted based on the transmission rates in the descending order, and PRBs are then allocated according to this order until each user's QoS requirement is satisfied.

The third block of G-SEM is a masking mechanism, which resolves the dependence among BSs with different colors. Taking one cell as an example, we define free PRBs for each link between a user and its associated BS:

20001372

Algorithm 3 : Two-stage slice enforcement algorithm.

```
Input: The interference graph $\mathcal{G}_\mathrm{I}$, the long-term slicing policies $\mathcal{P}$.
Initialization: Initialize the actor parameters $\boldsymbol{\theta}_\mathrm{A}$, the critic parameters $\boldsymbol{\theta}_\mathrm{V}$, the number of episodes $K_\mathrm{ep}$,
    the number of time steps in each episode $T_\mathrm{s}$.
for  $k = 1$ to $K_\mathrm{ep}$ do
    Clear the replay buffer for the critic;
    for  $t = 1$ to $T_\mathrm{s}$ do
        The actor network generates the action, i.e., the estimation of SNR degradation: $\mathbf{a}(t) \leftarrow$ the action
            sampled by $\log p_{\boldsymbol{\theta}_\mathrm{A}}(\mathbf{a}|\mathbf{s})$;
        Convert the SNR degradation into transmission rates according to equation (3.6);
        Run the heuristic algorithm G-SEM to obtain PRB allocation results for the current time step;
        Collect the instantaneous reward $\tilde{r}(t)$ from the environment according to equations (3.25)(3.26)(3.27);
        Compute the state value $v \leftarrow V^\pi(\mathbf{s}, \mathcal{G}_\mathrm{I}; \boldsymbol{\theta}_\mathrm{V})$;
        Store the tuple $(\log p_{\boldsymbol{\theta}_\mathrm{A}}(\mathbf{a}|\mathbf{s}), \mathbf{s}, \mathcal{G}_\mathrm{I}, \mathbf{a}, v, r)$ in the replay buffer;
        Update the state $\mathbf{s}(t)$ to the next state $\mathbf{s}(t+1)$;
        Reset the gradients $d\boldsymbol{\theta}_\mathrm{A} \leftarrow 0, d\boldsymbol{\theta}_\mathrm{V} \leftarrow 0$.
    end for
    $\mathrm{Loss}_\mathrm{a} = \sum_{t=1}^{T} A_t^\pi \log p(\mathbf{a}(t)|\mathbf{s}(t))$;
    $\mathrm{Loss}_\mathrm{v} = \sum_{t=1}^{T} \left[\sum_{t'=t}^{T} \gamma^{t'-t}\tilde{r}(t') - V^\pi(\mathbf{s}(t))\right]^2$;
    Update $\boldsymbol{\theta}_\mathrm{A}$ using gradients wrt. $\mathrm{Loss}_a$;
    Update $\boldsymbol{\theta}_\mathrm{V}$ using gradients wrt. $\mathrm{Loss}_v$.
end for
if users' QoS requirements cannot be satisfied then
    Conduct user admission control, i.e., discard the unsatisfied users and start a new round of learning.
end if
Output: A well-trained PRB allocation scheme for slice enforcement.
```

free PRBs are those that can be allocated to the user conforming to the predetermined transmission rates and ensuring slice isolation at the same time. The key idea of the masking mechanism is to identify all the free PRBs for each user. For cell-center users which are interference-free, all the PRBs are counted as free PRBs. However, it is not the case with cell-edge users.

How to identify free PRBs for a particular cell-edge user is illustrated as follows. A cell-edge user's link is deemed as the central node for the following. The one-hop neighbors of the central node is traversed on the interference graph. If the neighbors are associated with different slices from the central node, we mask the PRBs that are already allocated to these neighbors, i.e., directly set the PRB indicators to zero. If the neighbors are with the same slice as the central node, the masking mechanism is more delicate. We check the interference level on each PRB for both the central node and its neighbors, to see whether the interference will exceed the maximum tolerance interference (equation (3.8)) on that PRB. This is equivalent to ensuring that there is no violation of the predetermined transmission rates. The PRB allocation indicator is masked to be zero if the interference on that PRB is intolerable. The masking mechanism identifies all the free PRBs that can be allocated in the second block of LinkSlice.

The whole procedure of G-SEM is shown in algorithm 2. The three blocks designed in G-SEM efficiently

20011372

solve the multi-cell slice enforcement problem with a polynomial time complexity, which is analyzed in theorem 3.3. By prioritizing links and PRBs, G-SEM reduces the number of PRBs as much as possible for each slice, and thus reduces the overall PRB usage and slicing policy violation. G-SEM achieves a near-optimal solution to problem P3, which is evaluated in Section 3.5.

3.4.3 Overall Algorithm of LinkSlice

As shown in algorithm 3, the overall algorithm of two-stage slice enforcement is presented. The DRL agent is trained with each episode equal to the slicing policy period T_s, and the next episode is obtained by sliding the entire policy period by one radio frame. During the training process of K_ep episodes, the actor observes the states and generates actions sampled for each sector. The action, i.e., estimated SNR degradation, is converted to transmission rates, which are the given inputs of the second stage. G-SEM algorithm is executed to obtain PRB allocation schemes. Rewards are derived based on PRB scheduling for each frame. The DRL algorithm collects the rewards to update actor and critic networks. After K_ep episodes of training, the two stages work together iteratively until reaching convergence. By then, an effective link-level slice enforcement tool is obtained.

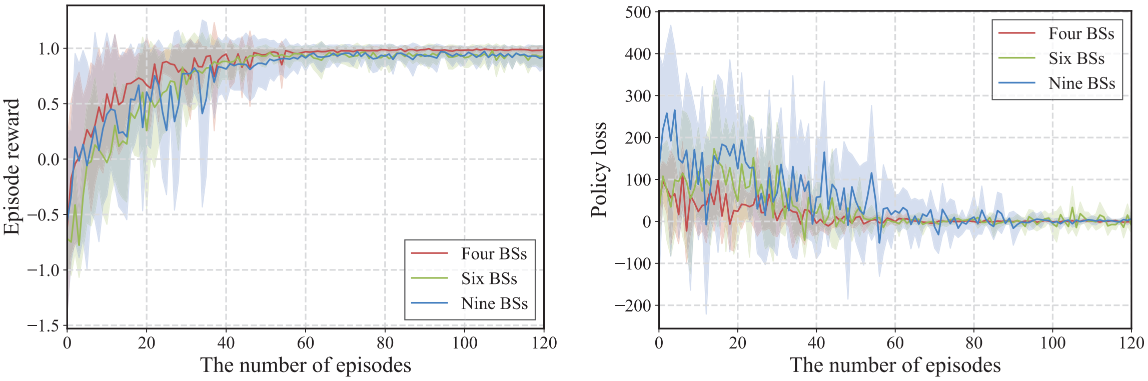
At the end of DRL training, if the QoS requirement of some users cannot be satisfied, then the network is considered overloaded, and user admission needs to be taken into account. In this case, the algorithm will automatically discard the users whose data rates are below their QoS requirements and start a new round of slice enforcement. This admission control part is incorporated into algorithm 3 for the integrity of LinkSlice.

3.5 Performance Evaluation

3.5.1 Simulation Setup

In this section, the performance of LinkSlice is evaluated via extensive simulations. A multi-cell RAN aligning with 3GPP standards (3GPP TR 38.801, 2017; 3GPP TR 38.802, 2017) is simulated. A frequency division duplexing (FDD) system is considered, and this chapter is focused on downlink scenarios. In the simulation, 20 MHz bandwidth is divided into 612 subcarriers with the subcarrier spacing set as 30 kHz. The radio frame size is 10 ms, consisting of 20 slots each with 0.5 ms. The total number of PRBs per slot is 51, and thus the number of PRBs within each radio frame is 1020 (i.e., $N_\mathrm{PRB} = 1020$). Since a centralized scheduling scheme is adopted, the interaction between the central controller and base stations incurs additional signaling overheads. The detailed analysis of overheads requires further investigation, which is not included in this chapter.

For a downlink channel from a BS to its associated users, the large-scale fading follows the UMi-Street



(a) Episode reward (b) Policy loss

Figure 3.4: Convergence performance of LinkSlice in the first 120 episodes with a different number of BSs.

canyon model in 3GPP TR 38.802 (2017), and the small-scale fading is Rician fading. The coherence time is assumed to be comparable with the length of one radio frame. There exists no frequency selectivity within one PRB's bandwidth. Modulation and coding schemes (MCS) are selected based on the standard link adaptation mechanism, and the CQI table and MCS index tables are specified in TS 38.214 (3GPP TS 38.214, 2021).

Users are distributed over each cell following uniform distribution unless specified differently. Users are differentiated by data rates (i.e., the metric of QoS requirement). In our simulations, four types of data rates, i.e., 100 kbps, 500 kbps, 1 Mbps, and 5 Mbps, are considered to represent services with low, medium, and high data rate requirements. User density is represented by the number of users per cell. By increasing user density, the traffic load in the network increases.

There are four types of slices in the simulations, and without loss of generality, one slice instance is considered in each type of slice. Within one slice instance, the user type is assumed to be homogeneous in data rates. Based on the previous analysis in section 3.1, the slicing policy on a BS, e.g. BS b, is represented by a four-dimension tuple, i.e., $\mathbf{p_b} = [p_{b,1}, p_{b,2}, p_{b,3}, p_{b,4}]$. Since slicing policies can be unbalanced, a parameter q_b called unbalance index is defined to quantify the level of unbalance as follows. Considering BS b as a central BS, the sum of Euclidean distances between its slicing policy vector $\mathbf{p}_b$ and that of its neighboring base stations is determined and then normalized. Thus, $q_b = \sum_{d\in\mathcal{N}(b,m)} \|\mathbf{p_b} - \mathbf{p_d}\| / (\sqrt{2}\,|\mathcal{N}(b,m)|)$, where $\mathcal{N}(b,m)$ is the set of neighboring base stations of BS b, and $|\mathcal{N}(b,m)|$ is the cardinality of the set. Apparently, a higher q_b indicates more unbalanced in slicing policies. Network-wide unbalance index is the average of all the base stations' unbalance indices. In the simulations, six different levels of network-wide unbalance index are considered, i.e., $0, 0.2, 0.4, 0.6, 0.8, 1.0$. The unbalance index is set to 0.8 if not specified explicitly. In subsection 3.5.5, the performance under various unbalanced policies is evaluated.

20011372

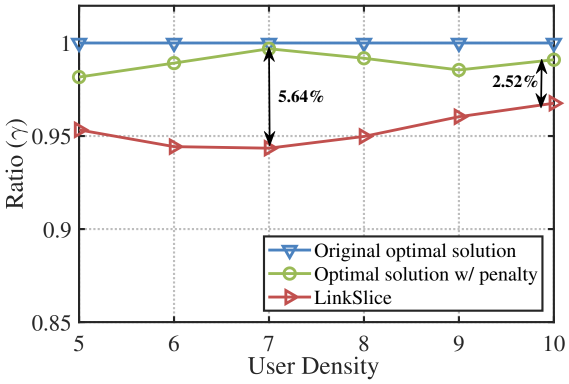


Figure 3.5: Comparisons with the optimal solutions to P1 and P2.

As explained in Section 3.4, the DRL agent is trained with each episode equal to the slicing policy period T_s, and the next episode is obtained by sliding the entire policy period by one radio frame. In the training process, $T = 10$, i.e., each episode corresponds to 100 ms of physical time in the network. However, considering two consecutive episodes, the overall time interval is only increased by 10 ms, i.e., one radio frame, because of the sliding operation of the policy period.

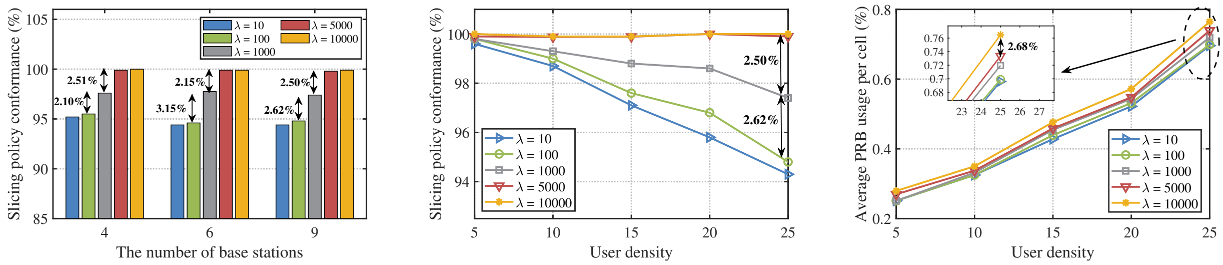
The hyper-parameters for DRL training are summarized as follows. The maximal number of episodes in the training process is set to 300. For the actor network design, the two-layer GCN that is used for state embedding has an input dimension of 53 (51 neurons for channel states per PRB in one frame, and 2 neurons for link rate and QoS satisfaction). The sizes of GCN layers are $\{53 \times 64\}$ and $\{64 \times 128\}$ respectively. The sizes of FCNN layers are $\{128 \times 64\}$ and $\{64 \times 16\}$. The final output dimension, i.e., action dimension d_a, is set as 16, which equals the number of SNR degradation levels. The reason is as follows. As stated in 3GPP specification TS 38.214 (3GPP TS 38.214, 2021), there are 16 CQI indices in all CQI tables, corresponding to 16 discrete SNR levels. If the degraded SNR level of a link falls out of the range of CQI table, the lowest transmission rate (i.e., MCS) is selected to serve that link. Finally, the SoftMax activation is used for generating the probability distribution of actions. For the critic network design, the only change is to modify the output dimension of FCNN as one. The learning rates for actor and critic networks are set as 0.0002 and 0.0001 respectively. The discount factor for accumulated rewards is set as 0.98.

Performance Metrics. To evaluate performance of LinkSlice, several performance metrics are considered. The first one is network throughput, which is defined as the total supported date rates of all users in the network. When QoS satisfaction of users is ensured, a slice enforcement scheme is considered better if its achieved network throughput is higher. A related performance metric is saturation throughput, which is obtained when the network is saturated, i.e., no more users can be admitted into the network. The second one is QoS satisfaction

20011372

Table 3.1: Key simulation parameters.

Parameter	Value	Parameter	Value
Large-scale fading	UMi-Street canyon	Fast fading	Rician
Carrier frequency	4 GHz	Subcarrier spacing	30 kHz
Bandwidth	20 MHz	The number of PRB	51/slot
MCS index table	(3GPP TS 38.214, 2021)	Base station	4,6,9
Frame size	10 ms	Slot size	0.5 ms



(a) Various number of base stations (b) Various user densities (c) Average PRB usage per cell

Figure 3.6: Slicing policy conformance and network performance with various penalty coefficients.

of users per slice, which is evaluated and compared between LinkSlice and other schemes. The third performance metric is slicing policy conformance. It is measured over one slicing policy period (i.e., 10 radio frames), and is then averaged over 1000 periods. The third metric is the number of reused PRBs across interfering links of different slices (i.e., interfering PRBs), which is used to quantify the performance of slice isolation. The smaller is this number, the better is slice isolation, and the best case is zero.

Other Schemes. To fully illustrate the performance of LinkSlice, it is compared with several other schemes. The first one is the optimal scheme. Its results are obtained directly from P1 by an optimization solver. The suboptimal scheme from P2 is also compared, and its results are also obtained from an optimization solver. The next scheme is random enforcement in which the required number of PRBs is selected randomly from a predetermined set. The users in the same cell are allocated resources sequentially. The state-of-the-art scheme to compare is the Most-Linked-First (MLF) scheme in D'Oro et al. (2019). It is designed based on the linked index that is defined as the amount of reused PRBs for a single slice across multiple cells. The slice-level enforcement is conducted so that the sum of linked indices is maximized.

3.5.2 Convergence of LinkSlice

The convergence of two-stage slice enforcement is shown in Fig. 3.4. Note that the solid line represents the mean value that is computed out of 500 runs, and the shaded area represents the standard deviation. For a

4-BS network, the episode reward increases to reach a plateau that is close to one after 60 training episodes, as shown in Fig. 3.4(a). Here the reward is normalized into the range $[-1, 1]$, which is a common practice in DRL (Henderson et al., 2018). Furthermore, as shown in Fig. 3.4(b), the policy loss decreases to reach a plateau that is close to zero after 60 training episodes.

The DRL agent reaches convergence after 60 episodes of learning, where one episode represents 10 radio frames. Thus, when the DRL algorithm runs online, it can converge within 0.69 s (i.e., 100 ms + 59 × 10 ms). In a cellular network, user mobility does not lead to a fast change in the interference pattern between communication links. In other words, the interference graph remains unchanged for tens of seconds. Thus, the training time of LinkSlice is much shorter than inference time during which LinkSlice is applied to conduct slice enforcement. In addition, when the number of nodes increases (i.e., the number of base stations is set as 6 and 9 while the user density is fixed), the convergence time of LinkSlice is still around 60 to 65 episodes. As a result, LinkSlice is highly adaptive to different interference graphs.

3.5.3 Performance Compared with Optimal Solutions

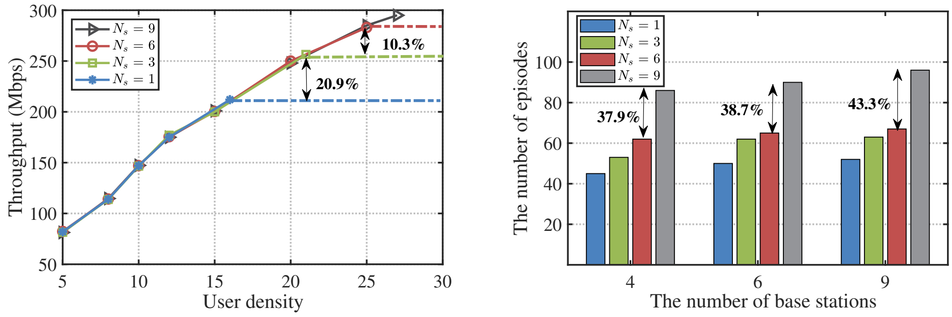
The performance of LinkSlice is compared with the optimal solution to problems P1 and P2. Both of the optimization problems, P1 and P2, are IQCP problems, which can be solved by Gurobi Optimizer, a business optimization solver designed for large-scale problems (Gurobi Optimization, LLC, 2021). For a 4-BS network, there are millions of integer variables in both P1 and P2. Gurobi Optimizer can obtain the optimal solution to such a problem but it takes several hours or even days. The minimal resource usage, i.e., the minimal number of PRBs, is compared among LinkSlice and the optimal solutions to problems P1 and P2. The parameter γ is defined as the ratio of the performance gap (in the minimal resource usage) as compared to the optimal objective value of problem P1.

For a 4-BS network, the comparison is conducted among the best solutions that can be achieved by tuning the penalty coefficient and other parameters. As shown in Fig. 3.5, the optimal solution with the penalty (P2) approaches the original optimal solution (P1) closely with a gap of less than 3%. Moreover, The largest gap for LinkSlice and the original optimal solution is around 5.6%, while the smallest gap is around 3.4%. This result indicates that LinkSlice can achieve a near-optimal result of slice enforcement.

3.5.4 Key Parameter Determination

Slicing Policy Conformance with Various Penalty Coefficients. Since LinkSlice is designed based on the transformed optimization problem P2 instead of the original optimization problem P1, the performance of

20011372



(a) Network throughput (b) Sample efficiency

Figure 3.7: Tradeoff between network throughput and sample efficiency with various sector sizes.

slicing policy conformance is closely related to the penalty coefficient λ in P2. To determine an appropriate value of the penalty coefficient, slicing policy performance is evaluated with respect to various penalty coefficients under a different number of BSs.

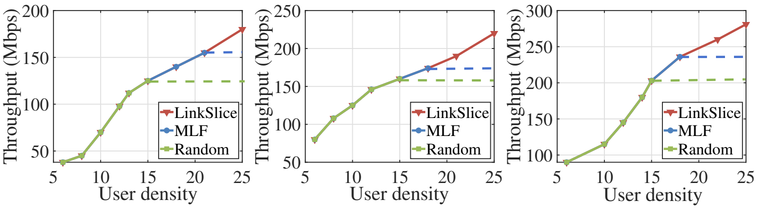
As the penalty coefficient varies from 10 to 10000, there is a tradeoff between slicing policy conformance and system performance in terms of PRB resource usage, which is shown in Fig. 3.6(b) and Fig. 3.6(c). The experiments are conducted in a 9-BS scenario with uniform user distribution, and the unbalance index is set as 0.8. When the penalty coefficient λ increases from 10 to 10000, the slicing policy conformance increases by 5.60% with the user density equal to 25. Especially, when the penalty coefficient reaches 5000 or above, 99.8% conformance is guaranteed over 1000 periods of slicing policy. However, more PRB resources are needed to fulfill the same user demands as the penalty coefficient increases. In other words, the spectrum efficiency is compromised with a large penalty coefficient. When the penalty coefficient reaches 10000, the network needs 6.85% more PRB resources per cell on average than that with $\lambda = 10$, and 2.68% more resources than that with $\lambda = 5000$. Furthermore, the impact of the number of users and base stations on the penalty coefficient determination is evaluated. As shown in Fig. 3.6(a), as the number of base stations increases, the slicing policy conformance with the same penalty coefficient varies within 1.2%. Thus, the impact of the number of base stations is negligible. As the user density increases, there exists a gap in the slicing policy conformance caused by various penalty coefficients, as shown in Fig. 3.6(b). Especially, when the user density is 25, the slicing policy conformance increases 5.6% as the penalty coefficients increases from 10 to 10000. Thus, to guarantee slicing policy conformance, a larger penalty coefficient needs to be used as the user density increases.

To balance the slicing policy conformance and system performance, the penalty coefficient is set to 5000 in the learning process*. In this case, the system ensures 99.8% slicing policy conformance, with a tolerable

---

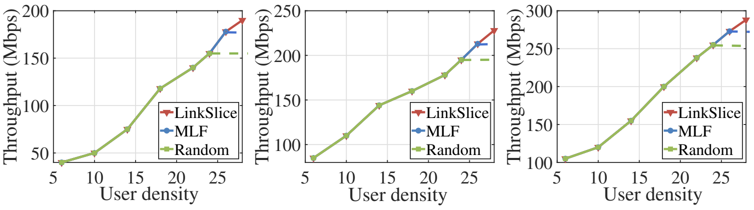
*In practice, at first, we choose the penalty coefficient λ so that the penalty term is 5 to 10 times larger than the average PRB usage per cell. Afterward, if the slicing policy conformance does not increase during the learning process, the penalty coefficient is tuned to a larger value, until the policy conformance is ensured.

20011372



(a) Four BSs (b) Six BSs (c) Nine BSs

Figure 3.8: Network throughput in the uniform user distribution.



(a) Four BSs (b) Six BSs (c) Nine BSs

Figure 3.9: Network throughput in the cell-center-biased user distribution.

decrease in the spectrum efficiency.

Graph Pooling with Various Sector Sizes. As mentioned in Section 3.4, GCNs are incorporated into the DRL agent to perform state embedding. To reduce the action space, a graph pooling mechanism is used to pool all embeddings within one sector as one. The sample efficiency of DRL algorithms is defined as the reciprocal of the number of episodes needed for DRL agents to reach the same performance.

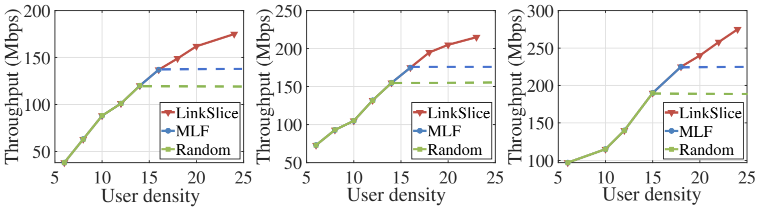
To evaluate the impact of the number of sectors per cell on the performance of LinkSlice, the tradeoff between the sample efficiency and network performance in terms of throughput is analyzed. As the number of sectors increases, the network throughput increases, while more episodes are needed for DRL convergence. Especially, as shown in Fig. 3.7(a), when the number of sectors N_s per cell is six, the network throughput is 30.3% higher than that with $N_s = 1$, and 4.2% lower than that with $N_s = 9$. As shown in Fig. 3.7(b), when the number of sectors is larger than 6, at least 37.9% more episodes are needed to train the DRL agent, but the performance improvement is limited. Therefore, in the following simulations, the number of sectors is set to 6.

3.5.5 Performance Compared with Existing Schemes

The performance of LinkSlice is evaluated and compared with other schemes in terms of network throughput, QoS satisfaction, slicing conformance, and slice isolation.

Network Throughput. Since user distribution impacts inter-slice interference of LinkSlice, three different user distribution scenarios are considered, i.e., uniform, cell-center-biased, and cell-edge-biased. In the uniform

20011372



(a) Four BSs (b) Six BSs (c) Nine BSs

Figure 3.10: Network throughput in the cell-edge-biased user distribution.

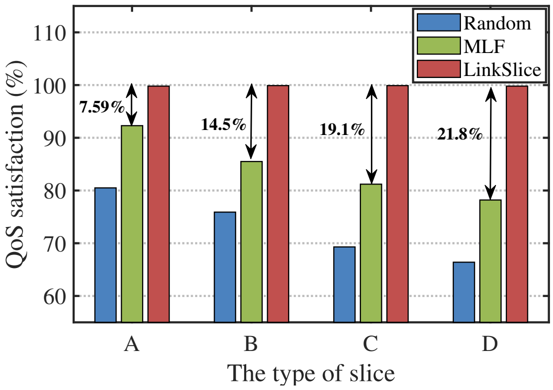


Figure 3.11: QoS satisfaction in the saturation scenario of LinkSlice.

scenario, users are uniformly distributed in the cell. For the cell-center-biased scenario, users are more densely distributed in cell-center regions, while for the cell-edge-biased scenario, users are more densely distributed in cell-edge regions.

We first consider the network throughput in the uniform user distribution scenario, with an unbalance index of 0.8 in the slicing policies. The multi-cell network goes from the unsaturated to saturated state as the user density increases from 5 to 25. The simulation results are illustrated in Figs. 3.8, 3.9, and 3.10 for the uniform, center-biased, and edge-biased scenarios. In all scenarios, the network throughput increases with the user density and reaches the plateau (i.e., saturation) at a certain user density. For ease of presentation, all curves are extended via a flat dash line to cover the same range of user density. As shown in these results, LinkSlice supports the highest user density when saturation starts, so it can achieve the highest saturation throughput as compared to the existing schemes, but the performance gain depends on user distribution. More specifically, in the uniform scenario, LinkSlice achieves a network throughput of 18.5% higher than the MLF scheme and 40.5% higher than the random enforcement. In the cell-center-biased scenario, however, the performance gain of LinkSlice is not as significant as that in the uniform case. The reason is that LinkSlice is advantageous at scheduling interfering links, but there exist few interfering links when most users are distributed in the cell-center

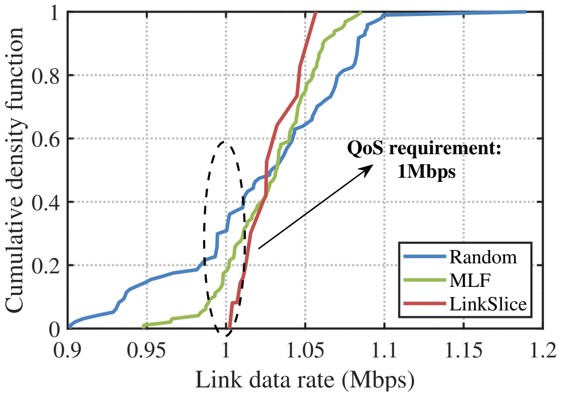


Figure 3.12: CDF plot of link data rate distribution for Slice C (1Mbps) in the saturation scenario of LinkSlice.

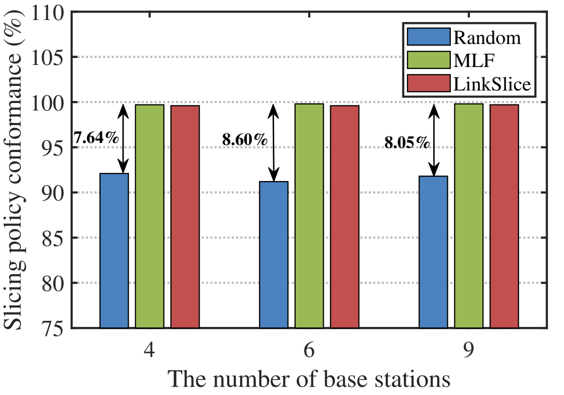


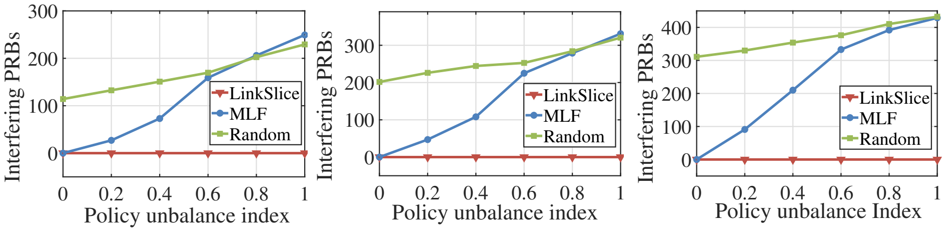
Figure 3.13: Comparisons of slicing policy conformance.

region.

QoS Satisfaction. The QoS satisfaction is compared among these three schemes under the highest user density that LinkSlice can support (i.e., saturation scenario). As shown in Fig. 3.11, the users' QoS satisfaction per slice is evaluated for four types of slices with the required data rates 100kbps, 500kbps, 1Mbps, and 5Mbps respectively (namely Slice A, B, C, and D). LinkSlice can well guarantee QoS requirements for the users associated with four different slices. Especially, the empirical CDF is used to demonstrate the average link data rate of users associated with slice C, as shown in Fig. 3.12.

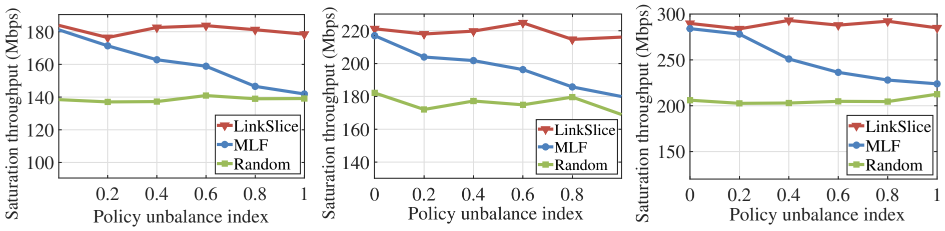
Slicing Policy Conformance. Slicing policy conformance is evaluated when the network reaches saturation. The long-term slicing policy conformance is compared among these three schemes. As is illustrated in Fig. 3.13, slicing policy conformance of LinkSlice reaches almost 99.8% conformance, while random enforcement can only guarantee 91.8% conformance. Since the MLF scheme is tailored to ensure slicing policy at the slice level instead of the link level, conformance to the slicing policies is not impacted by channel fading. Thus, it can always achieve high slicing policy conformance.

20011372



(a) Four BSs (b) Six BSs (c) Nine BSs

Figure 3.14: Number of interfering PRBs among different slices with various unbalance indices.



(a) Four BSs (b) Six BSs (c) Nine BSs

Figure 3.15: Saturation throughput under slicing policies with various unbalance indices.

Slice isolation. The relationship between slice isolation and the unbalance of slicing policies is revealed in Fig. 3.14. As slicing policies become more unbalanced, LinkSlice can always maintain nearly perfect slice isolation, i.e., the number of interfering PRBs among different slices is zero. However, for the random and the MLF schemes, it is difficult for them to ensure slice isolation under unbalanced slicing policies. The significant performance gain achieved by LinkSlice illustrates the advantage of link-level slice enforcement. As shown in Fig. 3.14, the MLF scheme achieves better isolation than the random scheme at a low unbalance index. However, when the unbalance index is higher than 0.8, the MLF scheme becomes worse and even the worst in slice isolation among all schemes. This result shows that a slice-level enforcement scheme like MLF is ineffective in ensuring slice isolation for unbalanced slicing policies.

Throughput under Unbalanced Slicing Policies Now the performance of LinkSlice under different slicing policies, either balanced or unbalanced, is evaluated. As is shown in Fig. 3.15, in all three multi-cell network scenarios, the MLF scheme cannot properly handle unbalanced slicing policies, while LinkSlice and random algorithm remain stable (within a small fluctuation range). As the unbalance index increases from 0 to 1, saturation throughput drops by 18.5% in the MLF scheme.

20011372

3.6 Summary

A link-level slice enforcement scheme called LinkSlice was developed for network slicing in a multi-cell radio access network. It satisfies the key requirements of network slicing, i.e., slice isolation, QoS guarantees, long-term conformance to slicing policies. As compared to the state-of-the-art, LinkSlice is distinct with the capability of achieving high throughput and perfect slice isolation under unbalanced slicing policies. As link-level slice enforcement needs to handle channel fading and interference between communication links, DRL and GCN were leveraged to design LinkSlice. GCN helped capture the interactions among interfering links. By embedding GCN into its agent and then working together with an optimization block, DRL accomplished resource allocation for fine-grained slice enforcement through a training process. Since LinkSlice can be trained within a time interval that is much shorter than the changing period of interference graph or user demands, it can be applied online effectively.

20011372

Chapter 4

Self-Supervised Learning Assisted Online Adaptation of Neural Channel Estimators

A pretrained neural channel estimator cannot generalize to all channel environments, necessitating online adaptation. Conventional methods demand ground-truth channel coefficients as supervised labels, but such labels are unavailable online. To this end, a self-supervised task is introduced on top of the original channel-estimation task to facilitate label-free adaptation of neural estimators. Specifically, this task randomly masks a fraction of resource elements in each received frame and reconstructs such masked parts. To enable effective reconstruction, the task input must incorporate two components: the unmasked parts and estimated data-symbols of masked parts. These estimated symbols are obtained via an online symbol-recovery mechanism, so no additional pilot overhead is incurred. To consolidate the self-supervised task with the original task, a two-branch masked auto-encoder (MAE) model called ChannelMAE is developed, with each branch dedicated to one task. The two branches share the same encoder but use separate decoders. During online adaptation, the encoder is updated by optimizing the self-supervised branch, which learns channel statistical features and shares them with the channel-estimation branch. Therefore, online channel-estimation accuracy is much improved. Extensive experiments show that ChannelMAE reduces channel-estimation error by up to 71.8% and 87.1% compared with the pretrained model and the state-of-the-art adaptation scheme, respectively.

20011372

4.1 System Model

The Single-Input Single-Output (SISO) OFDM communication is considered with one receive antenna and one transmit antenna. One OFDM frame spans a transmission time interval (TTI) and includes N_s OFDM symbols and N_f subcarriers. Each resource element (RE) has one symbol time and one subcarrier. Let $\mathbf{Y}, \mathbf{H}, \mathbf{N} \in \mathbb{C}^{N_\mathrm{s} \times N_\mathrm{f}}$ be the received signals, the channel coefficients, the zero-mean additive white Gaussian noise of one OFDM frame, respectively. The transmit signal matrix is $\mathbf{X} \in \mathbb{C}^{N_\mathrm{s} \times N_\mathrm{f}}$. The frequency-domain received signal at the receive antenna during one TTI is $\mathbf{Y} = \mathbf{H} \odot \mathbf{X} + \mathbf{N}$, where $\mathbf{N} \sim \mathcal{CN}(0, \sigma_n^2)$ and σ_n is the standard deviation of Gaussian noise. Let N_sp be the number of OFDM symbols carrying pilots (i.e., pilot symbol time) within one frame. At each pilot symbol time, N_fp subcarriers carry pilots (i.e., pilot subcarriers). Thus, in total $N_\mathrm{sp}N_\mathrm{fp}$ REs are occupied with pilot symbols. With transmitted pilots $\mathbf{X}_\mathrm{p} \in \mathbb{C}^{N_\mathrm{sp} \times N_\mathrm{fp}}$, received pilots $\mathbf{Y}_\mathrm{p} \in \mathbb{C}^{N_\mathrm{sp} \times N_\mathrm{fp}}$ are obtained. Pilot-based Least-Square (LS) estimate $\hat{\mathbf{H}}_\mathrm{p} \in \mathbb{C}^{N_\mathrm{sp} \times N_\mathrm{fp}}$ is computed as:

$$\hat{\mathbf{H}}_\mathrm{p} = \underset{H_\mathrm{p} \in \mathbb{C}^{N_\mathrm{sp} \times N_\mathrm{fp}}}{\arg\min} \|\mathbf{Y}_\mathrm{p} - \mathbf{H}_\mathrm{p} \odot \mathbf{X}_\mathrm{p}\|^2 = \frac{\mathbf{Y}_\mathrm{p}}{\mathbf{X}_\mathrm{p}}, \tag{4.1}$$

where the division between $\mathbf{Y}_\mathrm{p}$ and $\mathbf{X}_\mathrm{p}$ is element-wise.

Among conventional methods, linear Minimum Mean-Squared Error (LMMSE) channel estimator is widely acknowledged as a strong baseline, which estimates channel coefficients by linearly filtering $\hat{\mathbf{H}}_\mathrm{p}$ with channel cross- and auto-correlation matrices (Liu et al., 2014). But it is extremely challenging to obtain accurate channel correlations in practice. By contrast, neural channel estimators aim to learn a non-linear mapping from $\hat{\mathbf{H}}_\mathrm{p}$ to $\mathbf{H}$ using NNs in a purely data-driven fashion, without requiring prior channel statistics. This resolves the challenge of LMMSE estimators.

4.2 Overview of Two-Task Learning Framework

4.2.1 Task Formulation

As depicted in Fig. 4.1, the SSL and channel-estimation tasks (i.e., main task) are consolidated to a two-branch MAE model ChannelMAE, wherein they: 1) share one common feature encoder denoted by $\boldsymbol{\alpha}_\mathrm{e}$ and have their task-specific decoders (main-task decoder $\boldsymbol{\beta}_\mathrm{m}$ and SSL-task decoder $\boldsymbol{\beta}_\mathrm{s}$) and 2) they have different input-output pairs. We denote the main branch as $\boldsymbol{\theta}_\mathrm{m} = \{\boldsymbol{\alpha}_\mathrm{e}, \boldsymbol{\beta}_\mathrm{m}\}$ and the SSL branch as $\boldsymbol{\theta}_\mathrm{s} = \{\boldsymbol{\alpha}_\mathrm{e}, \boldsymbol{\beta}_\mathrm{s}\}$.

20011372

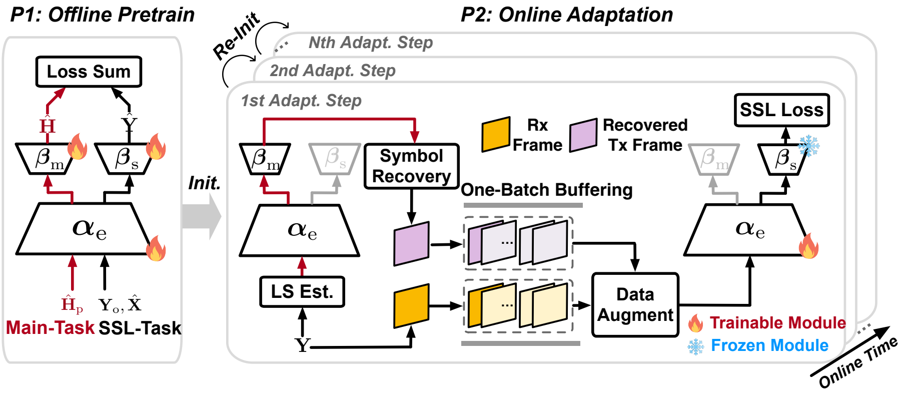


Figure 4.1: Overview of two learning phases.

Main-Task Formulation

The main branch takes pilot-based LS estimates $\hat{\mathbf{H}}_\mathrm{p}$ as the input and outputs full-frame estimated channel coefficients $\hat{\mathbf{H}}$. By denoting the mapping function parameterized by $\boldsymbol{\theta}_\mathrm{m}$ as $\varphi_{\boldsymbol{\theta}_\mathrm{m}}(\cdot)$, the main task is formulated as

$$\hat{\mathbf{H}} = \varphi_{\boldsymbol{\theta}_\mathrm{m}}(\hat{\mathbf{H}}_\mathrm{p}). \tag{4.2}$$

SSL-Task Formulation

The SSL task is formulated as reconstructing received frame $\mathbf{Y}$ from its randomly-masked version. Due to the randomness of transmitted data-symbols $\mathbf{X}$, directly reconstructing $\mathbf{Y}$ is infeasible. Thus, the estimated data-symbols $\hat{\mathbf{X}}$ must be incorporated into the task input as prior information to enable effective reconstruction. $\hat{\mathbf{X}}$ is obtained via an online symbol-recovery mechanism, as elaborated in Section 4.3.3.

Let $\mathbf{M}_\mathrm{r} \in \mathbb{R}^{N_\mathrm{s} \times N_\mathrm{f}}$ denote a random binary mask, where 1 indicates an unmasked position and 0 indicates a masked position. Following the masked-reconstruction paradigm, let $\mathbf{Y}_\mathrm{o} \in \mathbb{C}^{N_\mathrm{s} \times N_\mathrm{f}}$ be the masked frame after applying $\mathbf{M}_\mathrm{r}$, i.e., $\mathbf{Y}_\mathrm{o} = \mathbf{Y} \odot \mathbf{M}_\mathrm{r}$. The SSL branch takes $\mathbf{Y}_\mathrm{o}$ and $\hat{\mathbf{X}}$ as two raw inputs and reconstructs received frame $\hat{\mathbf{Y}}$ as the output. Letting $\phi_{\boldsymbol{\theta}_\mathrm{s}}(\cdot)$ be the mapping parameterized by $\boldsymbol{\theta}_\mathrm{s}$, the SSL task is expressed by

$$\hat{\mathbf{Y}} = \phi_{\boldsymbol{\theta}_\mathrm{s}}\left(\mathbf{Y}_\mathrm{o}, \hat{\mathbf{X}}\right). \tag{4.3}$$

4.2.2 Overview of Two Learning Phases

There exist two learning phases for ChannelMAE: offline pretraining and online adaptation. During offline pretraining, labeled data for both the main and SSL tasks are generated, and all parameters $\{\boldsymbol{\alpha}_\mathrm{e}, \boldsymbol{\beta}_\mathrm{m}, \boldsymbol{\beta}_\mathrm{s}\}$ are optimized through jointly learning the two tasks. During online adaptation, only the shared feature encoder $\boldsymbol{\alpha}_{\mathrm{e}}$ is adapted through SSL-task training, while the other parts of ChannelMAE remain frozen. The main branch is not trained as the supervised labels (i.e., true channel coefficients) are not available online.

As illustrated in Fig. 4.1, Online adaptation is conducted step-by-step, with each step spanning ΔN TTIs. At each TTI, the following procedures are performed: 1) the receiver obtains received frame $\mathbf{Y}$ and performs LS estimation to obtain $\hat{\mathbf{H}}_{\mathrm{p}}$ based on Eq. (4.1), and feeds $\hat{\mathbf{H}}_{\mathrm{p}}$ into the main branch to output $\hat{\mathbf{H}}$; 2) the receiver then recovers transmitted data-symbols $\hat{\mathbf{X}}$; 3) $\{\mathbf{Y}, \hat{\mathbf{X}}\}$ of the current TTI is stored in an online buffer. As ΔN TTIs of data are buffered, they are augmented into one online batch, over which the SSL-branch loss is computed and used to update $\boldsymbol{\alpha}_{\mathrm{e}}$. The buffer is then emptied, and the next adaptation step begins with $\boldsymbol{\alpha}_{\mathrm{e}}$ re-initialized to its updated parameters. Through this *batch-wise online learning* pattern, ChannelMAE is gradually adapted to new channel distributions over time with a low memory footprint in terms of data storage.

4.3 Key Designs of ChannelMAE

4.3.1 Input Pre-Processing

The shared feature encoder is realized by a transformer encoder. Thus, before entering the encoder, the inputs of the SSL task must be pre-processed and tokenized, while the input of the main task must be tokenized. Note that the complex tensors are transformed to real-valued ones by stacking their real and imaginary parts, which gives $\hat{\mathbf{H}}_{\mathrm{p}} \in \mathbb{R}^{N_{\mathrm{sp}} \times N_{\mathrm{fp}} \times 2}$ and $\mathbf{Y}, \mathbf{H}, \mathbf{X} \in \mathbb{R}^{N_{\mathrm{s}} \times N_{\mathrm{f}} \times 2}$. The last dimension of real tensors is termed ‘channel’ in this chapter.

SSL-Task Pre-processing: Random Masking and Input Fusion

Before converting the input of the SSL task into a sequence of tokens, both random masking and input fusion must be carefully designed. First, the random masking scheme, i.e., the procedure for generating $\mathbf{M}_{\mathrm{r}}$, is developed. As illustrated in Fig. 4.2(b), two variants are investigated: random-symbol masking and random-RE masking. In random-symbol masking, N_{rm} OFDM symbols in each frame are masked, leaving an unmasked part with dimensions $(N_{\mathrm{s}} - N_{\mathrm{rm}}) \times N_{\mathrm{f}}$. By contrast, the random-RE masking scheme randomly masks N_{re} REs in a frame. Random-symbol masking better preserves inter-symbol temporal coherence and thus results in a higher reconstruction accuracy, so it is employed in ChannelMAE.

Second, as the model simultaneously ingests $\mathbf{Y}_{\mathrm{o}}$ and $\hat{\mathbf{X}}$ as defined in Eq. (4.3), an input fusion mechanism is required to transform these raw inputs into a single composite input. We compare two fusion schemes as

20011372

in Fig. 4.2(b). The first scheme, termed concatenation-based fusion, concatenates the two inputs channel-wise, leading to $[\mathbf{Y}_\mathrm{o}; \hat{\mathbf{X}}]$. The second scheme computes an element-wise Hadamard quotient, $\mathbf{R}_\mathrm{o} = \mathbf{Y}_\mathrm{o} \odot \hat{\mathbf{X}}^{-1}$, which is called termed ratio-based fusion. In practice, the ratio-based fusion reduces input dimensionality and computational cost while delivering comparable performance to the other fusion scheme, so it is adopted in ChannelMAE.

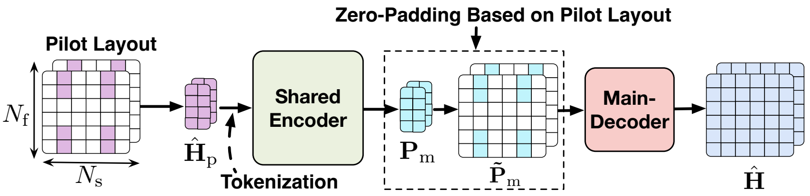
As depicted in Fig. 4.2(b), given the random-symbol masking and the ratio-based fusion schemes, the resulting composite input $\mathbf{R}_\mathrm{o} \in \mathbb{R}^{N_\mathrm{s} \times N_\mathrm{f} \times 2}$ is then tokenized, which will be elaborated later. Note that only its non-zero part denoted by $\breve{\mathbf{R}}_\mathrm{o} \in \mathbb{R}^{(N_\mathrm{s} - N_\mathrm{rm}) \times N_\mathrm{f} \times 2}$ is tokenized and sent to the encoder, thereby reducing the shared encoder's computation cost.

Input Tokenization for Both Tasks

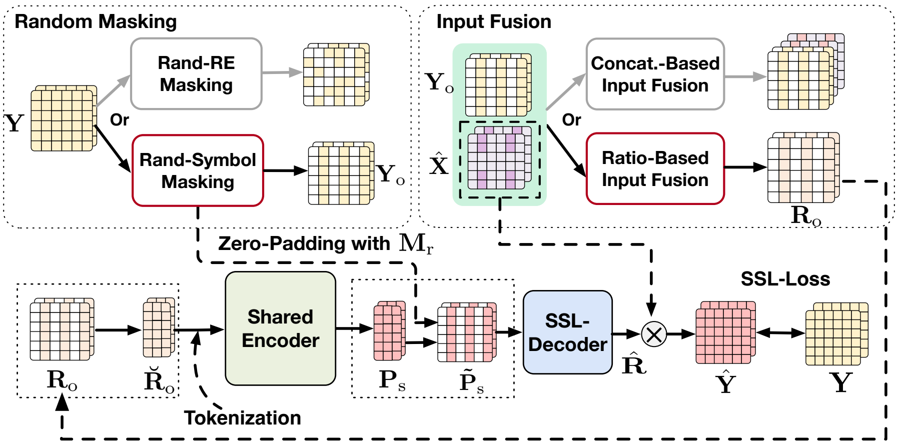
Main-task input $\hat{\mathbf{H}}_\mathrm{p} \in \mathbb{R}^{N_\mathrm{sp} \times N_\mathrm{fp} \times 2}$ is converted into a sequence of tokens $\{\mathbf{T}_\mathrm{m}\}$, letting $\mathbf{T}_\mathrm{m}$ be a single token of main task. Two tokenization schemes are compared. The first one is symbol-wise tokenization: it treats each pilot symbol time as one token, i.e., $\mathbf{T}_\mathrm{m} \in \mathbb{R}^{2N_\mathrm{fp}}$, with N_sp tokens in total. The other one is channel-wise tokenization that aggregates all values belonging to the same complex channel into a single token, producing only two tokens—one for the real part and one for the imaginary part—each of size $\mathbf{T}_\mathrm{m} \in \mathbb{R}^{N_\mathrm{sp} N_\mathrm{fp}}$. This tokenization allows the attention layer to capture time–frequency patterns over the entire frame more effectively, and thus the model performance outperforms the former one. Ablation studies further confirm that the channel-wise scheme consistently outperforms the symbol-wise alternative. Therefore, ChannelMAE uses channel-wise tokenization throughout its operation. Similar to the main-task tokenization, $\tilde{\mathbf{R}}_\mathrm{o}$ is converted into a sequence of two tokens, each of size $\mathbf{T}_\mathrm{s} \in \mathbb{R}^{(N_\mathrm{s} - N_\mathrm{rm}) N_\mathrm{f}}$, where $\mathbf{T}_\mathrm{s}$ represents a single SSL token.

4.3.2 Model Architecture: Two-Branch MAE

Both the main and SSL branches of ChannelMAE adopt an MAE architecture, as shown in Fig. 4.2. A transformer encoder is shared between two branches, and each branch ends in a ResNet-based decoder. Leveraging self-attention, the transformer encoder can learn global latent representation that CNN-based encoders often overlook (Vaswani et al., 2017). Moreover, the encoder in ChannelMAE only processes the non-zero parts of each input. Then, through zero-padding, the positional information of the pilot layout and the random mask is re-inserted into the main and SSL branches before entering the decoder, respectively, as shown in Fig. 4.2(a) and (b). Insertion of such information explicitly informs the model which parts are missing, enabling accurate reconstruction. The model components are detailed in the following.



(a) Detailed design of the main branch



(b) Detailed design of the SSL branch

Figure 4.2: Detailed designs of two model branches.

Shared Encoder

As shown in Fig. 4.3(a), the encoder $\boldsymbol{\alpha}_\mathrm{e}$ comprises N_e transformer layers (Vaswani et al., 2017), each containing a multi-head self-attention block and a two-layer MLP. Residual connections and layer normalization (LN) follow both blocks. Within each layer, self-attention captures correlations among input tokens. Letting $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ be queries, keys, and values, we have $\mathbf{Q} = \mathbf{K} = \mathbf{V}$. Throughout the encoder, their dimensions are kept as $\mathbb{R}^{N_\mathrm{seq} \times N_\mathrm{m}}$, where N_seq is the token count in the input sequence and N_m is the embedding dimension. Let $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{N_\mathrm{m} \times d_k}$ be the learnable parameters of each attention head i associated with queries, keys, and values, respectively, where d_k is the key dimension (also the query and value dimension). The output projection matrix $\mathbf{W}^O \in \mathbb{R}^{h d_k \times N_\mathrm{m}}$, where h is the number of attention heads. To this end, multi-head self-attention layer (Vaswani et al., 2017) $\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ is computed as

$$\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathrm{head}_1, ..., \mathrm{head}_h)\mathbf{W}^O \tag{4.4}$$

$$\mathrm{head}_i = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{W}_i^Q(\mathbf{K}\mathbf{W}_i^K)^\mathrm{T}}{\sqrt{d_k}}\right)\mathbf{V}\mathbf{W}_i^V, \tag{4.5}$$

20011372

where Softmax(·) denotes SoftMax function. The subsequent MLP has hidden size N_{hid} and output size N_{m}, with GeLu activation applied only after the first layer.

Note that tokens $\{\mathbf{T}_{\text{m}}\}$ or $\{\mathbf{T}_{\text{s}}\}$ stated in Section 4.3.1 are linearly projected to $\mathbb{R}^{N_{\text{seq}} \times N_{\text{m}}}$ and serve as the initial $\mathbf{Q}, \mathbf{K}, \mathbf{V}$. Since channel-wise tokenization is employed, $N_{\text{seq}} = 2$ for both tasks.

Main-Task Decoder

The output sequence of the encoder denoted by $\mathbf{P}_{\text{m}} \in \mathbb{R}^{2 \times N_{\text{sp}} N_{\text{fp}}}$ is reshaped to $\mathbb{R}^{N_{\text{sp}} \times N_{\text{fp}} \times 2}$. Using the pre-known pilot pattern, non-pilot positions are padded with zeros to obtain a full-size representation $\tilde{\mathbf{P}}_{\text{m}} \in \mathbb{R}^{N_{\text{s}} \times N_{\text{f}} \times 2}$, as shown in Fig. 4.2(a). This tensor passes through the main-task decoder that consists of an input convolutional layer, N_{dm} ResNet blocks (Lim et al., 2017), and an output convolutional layer, resulting in estimated channel coefficients $\hat{\mathbf{H}} \in \mathbb{R}^{N_{\text{s}} \times N_{\text{f}} \times 2}$.

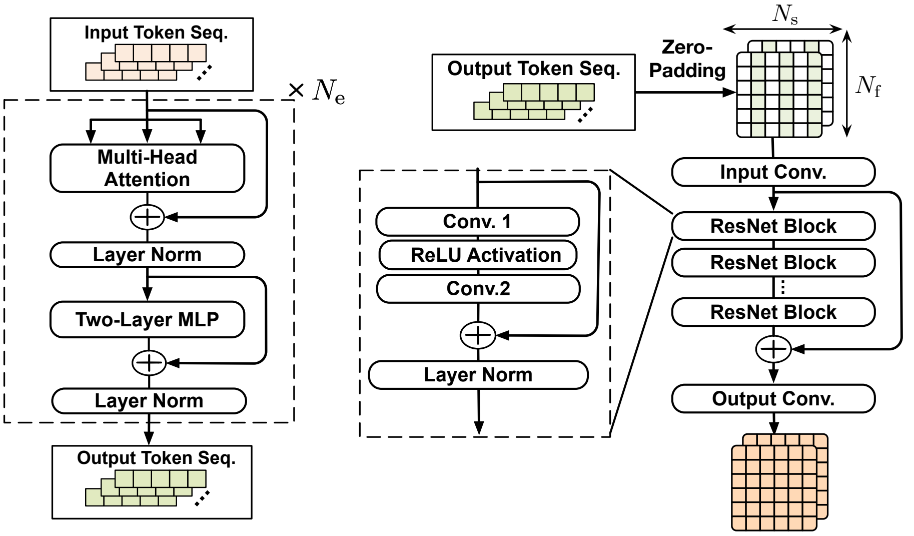
SSL-Task Decoder

The encoder output for the SSL branch, denoted as $\mathbf{P}_{\text{s}} \in \mathbb{R}^{2 \times (N_{\text{s}} - N_{\text{rm}}) N_{\text{f}}}$, is reshaped to $\mathbb{R}^{(N_{\text{s}} - N_{\text{rm}}) \times N_{\text{f}} \times 2}$. Then random-symbol mask $\mathbf{M}_{\text{r}}$ is inserted back to this representation, which gives $\tilde{\mathbf{P}}_{\text{s}} \in \mathbb{R}^{N_{\text{s}} \times N_{\text{f}} \times 2}$. The SSL-task decoder follows the same design as that of main-task decoder, except that it has a different number of ResNet blocks denoted by N_{ds}. Since only the SSL task is trained online to update the shared encoder, a more lightweight SSL-decoder is designed to reduce the back-propagation cost online (i.e., $N_{\text{ds}} < N_{\text{dm}}$). At last, as the ratio-based fusion is applied during pre-processing, the output of SSL-decoder denoted by $\hat{\mathbf{R}}$ is further post-processed to obtain reconstructed frame $\hat{\mathbf{Y}}$, i.e., $\hat{\mathbf{Y}} = \hat{\mathbf{R}} \odot \hat{\mathbf{X}}$, as depicted in Fig. 4.2(b).

4.3.3 Two-Phase Training Procedures

ChannelMAE is trained in two phases: offline pretraining and online adaptation. Let $\ell_{\text{m}}(\cdot)$ and $\ell_{\text{s}}(\cdot)$ represent per-sample loss functions of the main and SSL tasks, respectively. We denote $\mathbf{Y}'$ as the masked parts of $\mathbf{Y}$ and $\hat{\mathbf{Y}}'$ as the reconstructed masked parts extracted from $\hat{\mathbf{Y}}$. The SSL-task loss computes the squared error *only* on the masked parts of the frame, which is $\ell_{\text{s}}\big(\mathbf{Y}_{\text{o}}, \hat{\mathbf{X}}, \mathbf{Y}'\big) = \big\|\mathbf{Y}' - \phi'_{\boldsymbol{\theta}_{\text{s}}}(\mathbf{Y}_{\text{o}}, \hat{\mathbf{X}})\big\|_F^2$, where $\phi'_{\boldsymbol{\theta}_{\text{s}}}(\cdot)$ further applies the extraction of the masked parts upon $\phi_{\boldsymbol{\theta}_{\text{s}}}(\cdot)$, i.e., $\hat{\mathbf{Y}}' = \phi'_{\boldsymbol{\theta}_{\text{s}}}(\mathbf{Y}_{\text{o}}, \hat{\mathbf{X}})$, and $\|\cdot\|_F$ denotes the Frobenius norm. The main-task loss is also the squared error, i.e., $\ell_{\text{m}}\big(\hat{\mathbf{H}}_p, \mathbf{H}\big) = \big\|\mathbf{H} - \varphi_{\boldsymbol{\theta}_{\text{m}}}(\hat{\mathbf{H}}_p)\big\|_F^2$.

20011372



(a) Encoder (b) Decoder

Figure 4.3: Model architecture of ChannelMAE.

Offline Pretraining

ChannelMAE is first optimized offline, where the trainable parameters are $\boldsymbol{\theta} = \{\boldsymbol{\alpha}_\text{e}, \boldsymbol{\beta}_\text{m}, \boldsymbol{\beta}_\text{s}\}$. Both the ground-truth channel coefficients and transmit symbols are available in offline simulations, so $\mathbf{H}$ and $\mathbf{X}$ are known. Assuming N offline samples, the mean-squared error (MSE) loss is

$$\mathcal{L}_{\text{off}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\Big[\ell_{\text{m}}\big(\hat{\mathbf{H}}_{p}^{(i)}, \mathbf{H}^{(i)}\big) + \ell_{\text{s}}\big(\mathbf{Y}_{\text{o}}^{(i)}, \mathbf{X}^{(i)}, \mathbf{Y}'^{(i)}\big)\Big], \tag{4.6}$$

where i denotes the index of data sample. This loss is then minimized via multi-epoch training with respect to $\boldsymbol{\theta}$.

Online Adaptation

During online deployment, only the SSL task is learned to update the shared encoder. In each online adaptation step mentioned in Section 4.2, estimated data-symbols $\hat{\mathbf{X}}$ must be recovered after main-task inference. Let T be the index of online adaptation step and t be the index of TTI within each step. Thus $\boldsymbol{\alpha}_\text{e}^{(T)}$ represents the encoder used in online adaptation step T, and $\boldsymbol{\theta}_\text{m}^{(T)}, \boldsymbol{\theta}_\text{s}^{(T)}$ denote the main and SSL branch in the current step, respectively.

The online symbol-recovery mechanism is stated as follows. As shown in Fig. 4.4, two optional schemes are provided for recovering $\hat{\mathbf{X}}$, which are the uncoded and channel-coded recovery loops. First, in the uncoded recovery loop, the data symbols are detected by the conventional LMMSE symbol detector, and the output is

20011372

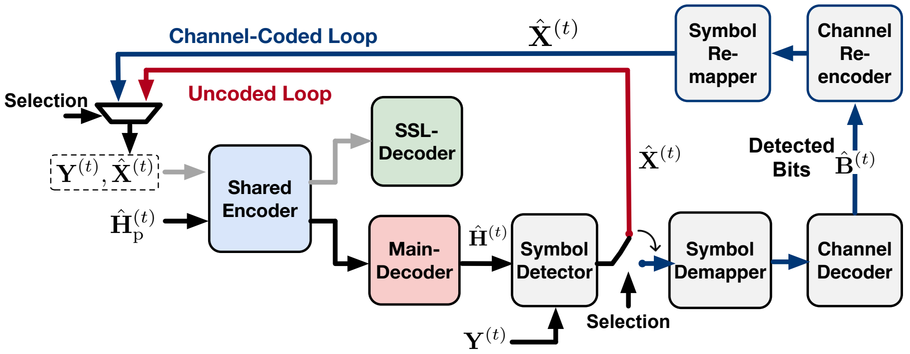


Figure 4.4: Uncoded and channel-coded online symbol-recovery schemes.

directly fed back as one of the input components of the SSL branch. This scheme conforms to the normal symbol-detection procedure and thus does not incur extra computation cost to the detection pipeline. Specifically, at TTI t, the main-task inference is performed through $\hat{\mathbf{H}}^{(t)} = \varphi_{\boldsymbol{\theta}_{\mathrm{m}}^{(T)}}(\hat{\mathbf{H}}_{\mathrm{p}}^{(t)})$, where $\hat{\mathbf{H}}^{(t)}, \hat{\mathbf{H}}_{\mathrm{p}}^{(t)}$ are the main-task input and output at TTI t. Given $\hat{\mathbf{H}}^{(t)}$, the LMMSE symbol detection is conducted via

$$\hat{\mathbf{X}}^{(t)} = f_{\mathrm{det}}(\hat{\mathbf{H}}^{(t)}, \mathbf{Y}^{(t)}), \tag{4.7}$$

where $f_{\mathrm{det}}(\cdot)$ characterizes the symbol detection process.

Second, in the channel-coded recovery loop, $\hat{\mathbf{X}}$ is recovered by incorporating channel decoding/encoding processes. In this scheme, we denote the detected bits of TTI t output by the channel decoder as $\hat{\mathbf{B}}^{(t)}$. Then the process of acquiring estimated symbols $\hat{\mathbf{X}}$ is:

$$\hat{\mathbf{B}}^{(t)} = g_{\mathrm{dec}}(\hat{\mathbf{H}}^{(t)}, \mathbf{Y}^{(t)}), \quad \hat{\mathbf{X}}^{(t)} = h_{\mathrm{enc}}(\hat{\mathbf{B}}^{(t)}), \tag{4.8}$$

where $g_{\mathrm{dec}}(\cdot)$ abstracts the receiving process of symbol detection, symbol de-mapping, and channel decoding; $h_{\mathrm{enc}}(\cdot)$ abstracts the additional process of channel re-encoding and symbol remapping. It is evident that the channel-coded recovery loop incurs extra high computation cost due to $h_{\mathrm{enc}}(\cdot)$. As validated in Section 4.4.3, both of the above schemes result in similar online adaptation performance. The online symbol-recovery mechanism adopts the uncoded loop because of its much lower computation cost.

At each TTI t, $\{\mathbf{Y}^{(t)}, \hat{\mathbf{X}}^{(t)}\}$ is pushed in the online buffer. During one online adaptation step, a batch of online samples $\{\mathbf{Y}^{(t)}, \hat{\mathbf{X}}^{(t)}\}_{t=1}^{\Delta N}$ with size ΔN is buffered. By the end of each step, this batch is popped and enlarged by an *augmentation factor* A. Specifically, for each $\mathbf{Y}^{(t)}$, A independent random masks are drawn and applied to generate A masked frames. Each original pair $\{\mathbf{Y}^{(t)}, \hat{\mathbf{X}}^{(t)}\}$ appears exactly A times in the augmented batch denoted by $\mathcal{D}_{\mathrm{aug}}$, yielding a total of $A\Delta N$ augmented training samples. $\mathcal{D}_{\mathrm{aug}}$ is the one-batch data for learning the SSL task online.

During online adaptation, the SSL decoder $\boldsymbol{\beta}_{\mathrm{s}}$ is frozen and only the shared encoder $\boldsymbol{\alpha}_{\mathrm{e}}$ is updated. Over the online augmented-batch data, the MSE loss is

$$\mathcal{L}_{\mathrm{on}}\big(\boldsymbol{\alpha}_{\mathrm{e}}\big) = \frac{1}{|\mathcal{D}_{\mathrm{aug}}|} \sum_{i=1}^{|\mathcal{D}_{\mathrm{aug}}|} \ell_{\mathrm{s}}\big(\mathbf{Y}_{\mathrm{o}}^{(i)}, \hat{\mathbf{X}}^{(i)}, \mathbf{Y}'^{(i)}\big). \tag{4.9}$$

The encoder parameters in the current adaptation step T are updated via one-step gradient descent to obtain new encoder parameters $\boldsymbol{\alpha}_{\mathrm{e}}^{(T+1)}$, i.e., $\boldsymbol{\alpha}_{\mathrm{e}}^{(T+1)} = \boldsymbol{\alpha}_{\mathrm{e}}^{(T)} - \mu \nabla_{\boldsymbol{\alpha}_{\mathrm{e}}} \mathcal{L}_{\mathrm{on}}\big(\boldsymbol{\alpha}_{\mathrm{e}}^{(T)}\big)$, with learning rate μ.

4.4 Performance Evaluation

4.4.1 Simulation Setup

Wireless System and Channel Datasets

Considering an SISO-OFDM uplink communication model, the system setup is as follows. The carrier frequency is 3GHz with subcarrier frequency 30kHz. Each OFDM frame consists of 14 symbols and 72 subcarriers, i.e, $N_{\mathrm{f}} = 72, N_{\mathrm{s}} = 14$. Within each frame, a 3GPP-aligned pilot pattern (3GPP TS 38.211, 2024) is used: two OFDM symbol times (the 3rd and 10th symbols) are selected as pilots, giving $N_{\mathrm{sp}} = 2, N_{\mathrm{fp}} = 72$. The modulation scheme is fixed as 4-QAM. LDPC channel coding employs code rate 658/1024.

Two categories of channel datasets are considered. First, 3GPP standard channel models, urban macro (UMa) and urban micro (UMi) (3GPP TS 38.901, 2024), are simulated. To demonstrate offline-online channel distribution shifts, UMa with low mobility (0–5 m/s) provides the offline distribution, whereas UMi with high mobility (25–30 m/s) serves online. Second, ray-traced (RT) real-world channels are generated with Sionna ray tracing tools (NVIDIA, 2025; Hoydis et al., 2023). RT-based channel datasets with mobility 0–10 m/s are prepared using five real-world city scenes (OpenStreetMap contributors, 2025), namely generic street canyon, Paris, Florence, Munich, and San Francisco, and they are labeled as City1–City5 for clarity. For a single distribution shift, we specifically employ City5 as the offline channel environment and City3 as the online one, and we also evaluate multiple online environment shifts.

Training Setup

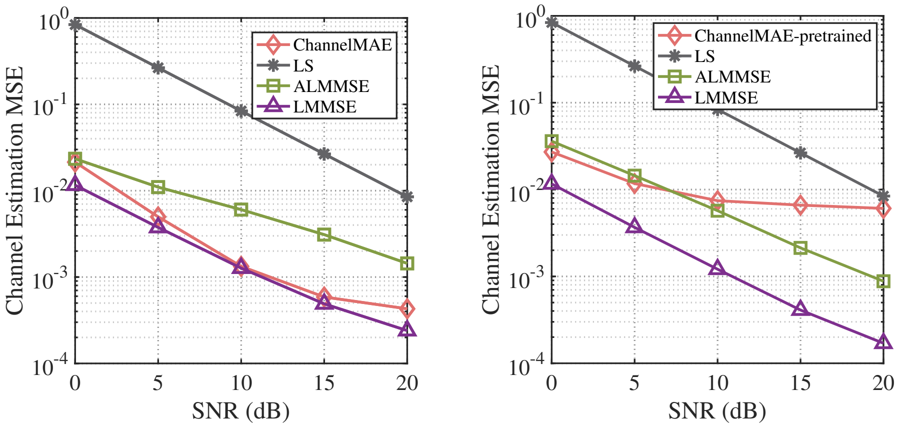
During offline pretraining, 40,000 TTIs in the SNR range of 10–20 dB are generated for each offline channel distribution, with the training/validation/test split as 0.8:0.1:0.1. After 80 training epochs using the Adam optimizer (with learning rate 0.001 and batch size 64). During online adaptation, the adaptation step contains 32 TTIs, and thus the original online batch size is also 32. The batch-wise online learning as stated in Section 4.3.3 is performed with learning rate 0.0005. In total 10,000 TTIs within SNR range 10–15 dB are simulated online. With the ultimate online-adapted model, we evaluate the online channel estimation performance with another separate test dataset. Unless otherwise specified, the key hyper-parameters are specified as follows: ChannelMAE uses one encoder layer ($N_\text{e} = 1$); $N_\text{dm} = 4$ and $N_\text{ds} = 2$ ResNet blocks in the main and SSL decoders, respectively; $N_\text{rm} = 12$ masked symbols in the SSL branch; data augmentation $A = 5$ times; an MLP hidden dimension of $N_\text{hid} = 16$; and an embedding dimension in the encoder $of N_\text{m} = 144$. Both decoders employ a kernel size of 5. All simulations are conducted with Sionna 1.0.1 (Hoydis et al., 2022) and TensorFlow 2.15.0 on an NVIDIA RTX 4090 and an Apple M4 CPU.

Baselines

Among conventional methods (Feriani et al., 2023), we include LS, ideal approximate-LMMSE (i.e., ALMMSE), and ideal LMMSE (i.e., LMMSE). The last two are both ideal baselines, as ALMMSE estimates channel statistics from true channel coefficients and LMMSE uses noiseless pilot estimates and perfect channel statistics for its implementation. For DL-based methods we compare ChannelMAE with a CNN-based benchmark ChannelNet (Soltani et al., 2019), the state-of-the-art transformer-based HA02 (Luan and Thompson, 2022, 2023), the self-supervised denoiser DnCNN (Zhang et al., 2023), the state-of-the-art model that supports online adaptation HA03 (Luan and Thompson, 2023; Kong et al., 2025). During online adaptation, only the last two models are considered as they can be adapted online without true channel coefficients. We also consider using ground-truth channel coefficients to train the main branch of ChannelMAE in a *supervised* manner, which serves as the upper bound for ChannelMAE if no data augmentation is applied.

To evaluate channel estimators, standard Monte-Carlo simulations are conducted at the selected SNR points, computing the MSE between estimated and ground-truth channel coefficients. The MSE gain (in dB) of scheme 2 over scheme 1 is defined as $10 \log_{10}(\text{MSE}_1/\text{MSE}_2)$, where MSE_1 and MSE_2 are MSEs of scheme 1 and 2, respectively. Computation cost is measured by TensorFlow Profiler in terms of floating-point operations in millions (MFLOPs).

20011372



(a) Offline RT channel (b) Online RT channel

Figure 4.5: Performance degradation of the pretrained model encountering the offline-online distribution shift.

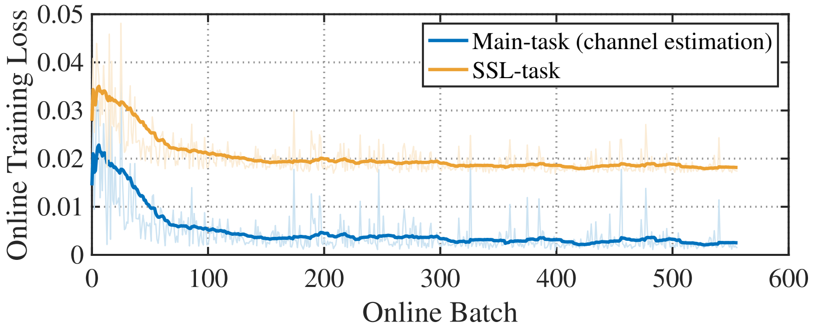
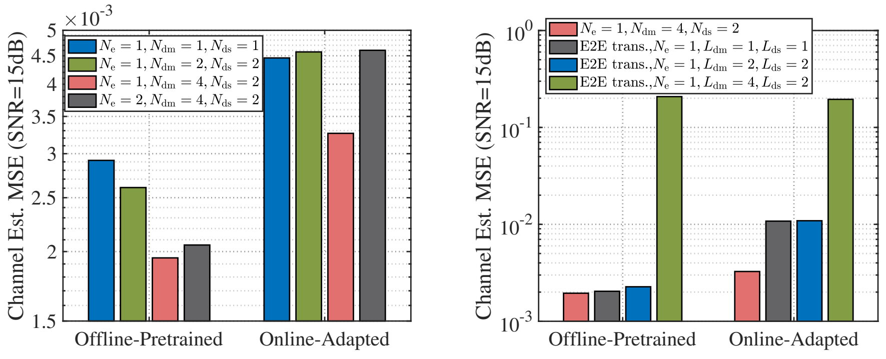


Figure 4.6: Online losses of ChannelMAE versus online batch index (both losses are running-averaged for 30 batches).

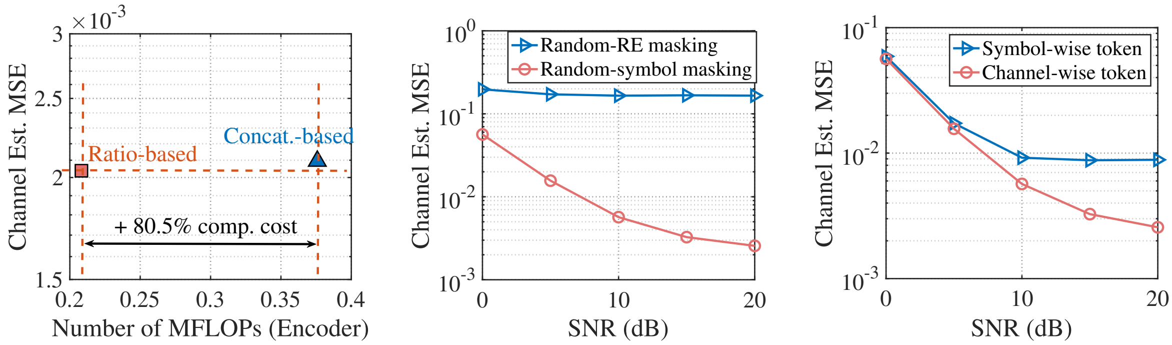
4.4.2 Validation of Online Performance Degradation and Online Model Convergence

ChannelMAE is pretrained in the offline RT channel and then evaluated in both the offline and online RT channels. As shown in Fig. 4.5(a), the pretrained model outperforms ALMMSE across 0–20 dB SNR and even achieves performance comparable to LMMSE. However, it witnesses a drastic performance degradation online, as shown in Fig. 4.5(b), which validates the necessity of online adaptation. The convergence behavior of online adaptation is depicted in Fig. 4.6, where the SSL task alone is trained to update the shared encoder. As more online batches are observed, the SSL-task loss decreases to a plateau, driving down the main-task loss as well. This indicates a clear synergy between the learning processes of the two tasks.



(a) Various numbers of layers (b) CNN vs. transformer decoder

Figure 4.7: Various model architectures of ChannelMAE.



(a) Input fusion (b) Masking scheme (c) Tokenization

Figure 4.8: Ablation studies of input pre-processing.

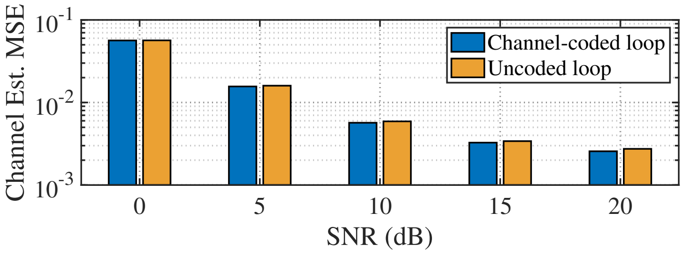
4.4.3 Ablation Studies

The following ablation studies are conducted using 3GPP channels and the determined designs are also applied to RT channels.

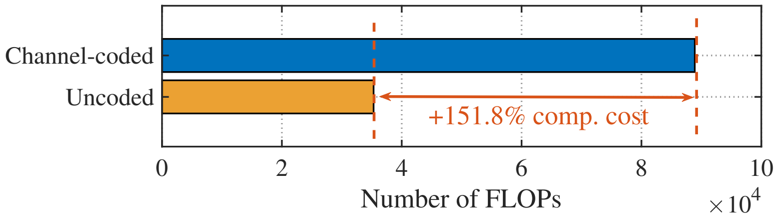
Model Architecture

The impact of various model architectures on ChannelMAE is studied in two aspects. First, the number of encoder layers (i.e., N_e) and the numbers of ResNet blocks in the main-decoder and SSL-decoder (i.e., N_dm, N_ds) are varied. The offline-pretrained and online-adapted models with various model structures are evaluated at an SNR of 15 dB, respectively. As shown in Fig. 4.7(a), the model with $N_\mathrm{e} = 1, N_\mathrm{dm} = 4, N_\mathrm{ds} = 2$ outperforms all others in both the offline and online phases. A relatively smaller SSL-decoder is designed to reduce online backpropagation cost through the SSL branch. Second, an end-to-end transformer-based MAE is compared with ChannelMAE. Specifically, the main and SSL decoders in ChannelMAE are replaced by transformer decoders

20011372



(a) Online symbol-recovery schemes



(b) Per-sample FLOP counts

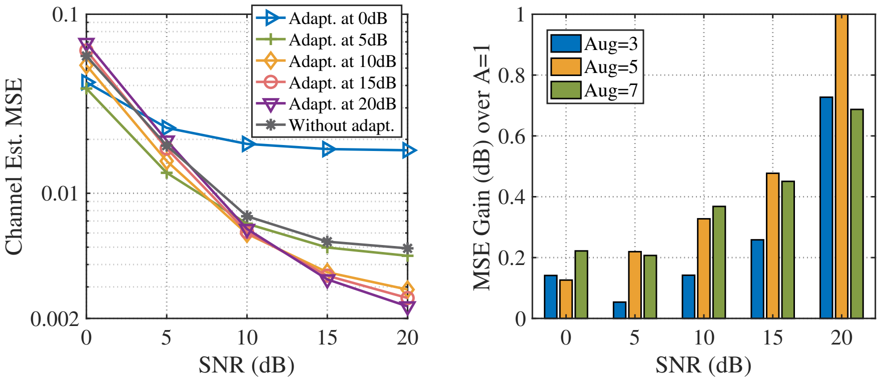
Figure 4.9: Comparison of two online symbol-recovery schemes in terms of channel estimation performance and computation cost.

(He et al., 2022) with L_{dm} and L_{ds} layers, respectively, while retaining the original encoder. This variant achieves a similar channel-estimation MSE to ChannelMAE when $L_{\mathrm{dm}} = L_{\mathrm{ds}} = 1$, but its performance degrades significantly during online adaptation as in Fig. 4.7(b). Based on these two analyses, the two-branch MAE is determined with $N_{\mathrm{e}} = 1$, $N_{\mathrm{dm}} = 4$, and $N_{\mathrm{ds}} = 2$.

Input Pre-Processing

As stated in Section 4.3.1, the following design aspects must be studied: 1) the random masking scheme and the input fusion scheme for the SSL-task; 2) the tokenization scheme for both tasks. We first compare two fusion schemes, ratio-based and concatenation-based fusion, in the pretraining phase. As shown in Fig. 4.8(a), the ratio-based fusion scheme achieves a slightly lower MSE than the concatenation-based scheme in the high-SNR region (i.e., 20 dB). Meanwhile, the latter one increases computation cost by around 80.5% in terms of FLOPs. Therefore, the ratio-based fusion scheme is adopted. Next, the random-symbol and random-RE masking schemes are compared. During online adaptation, random-RE masking fails to converge, producing a flat MSE curve, as seen in Fig. 4.8(b). Thus, random-symbol masking must be set for the SSL task. Last, two tokenization methods are compared in Fig. 4.8(c). The online-adapted model using channel-wise tokenization achieves an MSE gain of around 5.4 dB than the symbol-wise approach at 20 dB SNR, so the channel-wise tokenization is adopted.

20011372



(a) Online-adaptation SNR (b) Data augmentation

Figure 4.10: Comparison of various data augmentation times and online-adaptation SNR values.

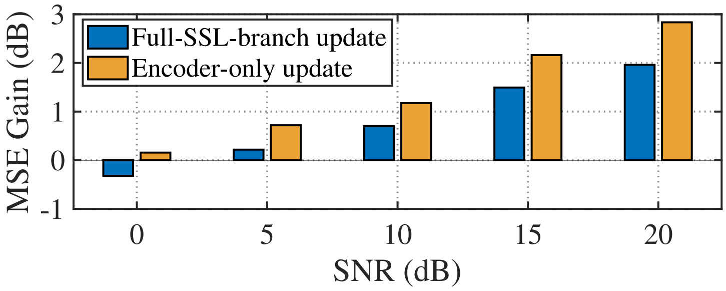


Figure 4.11: Comparison of online model updating strategies.

Online Symbol-Recovery Mechanism

Two schemes for obtaining estimated data-symbols are compared, as described in Section 4.3.3, depending on whether the channel coding procedures are included. As shown in Fig. 4.9, both schemes (channel-coded and uncoded loops) yield nearly identical performance across 0–20 dB SNR, indicating that the inclusion of channel coding does not benefit SSL-task learning. Moreover, using the channel-coded loop increases the number of FLOPs per online sample by 151.8%. Therefore, throughout the online adaptation process, we use the online symbol-recovery scheme with the uncoded loop.

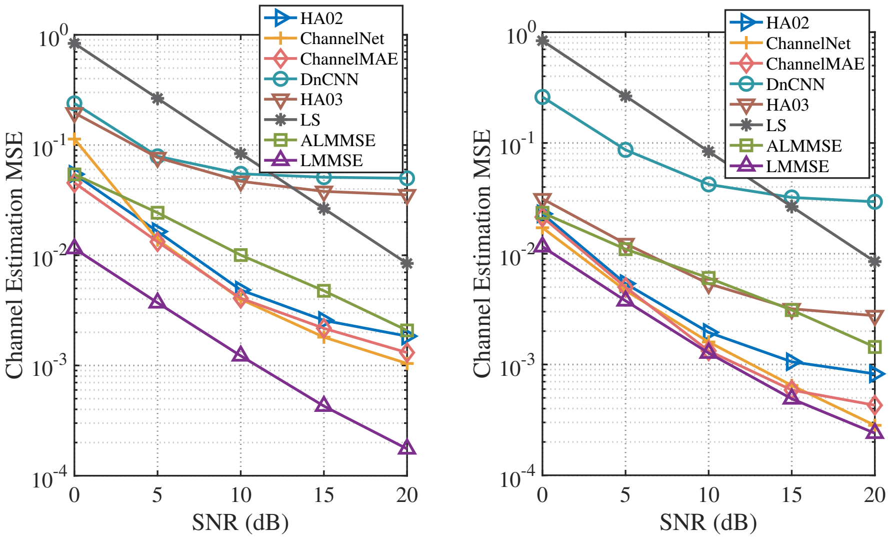
Online-Adaptation SNR and Data Augmentation

The impact of online-adaptation SNR (i.e., the SNR condition of online batches) on model performance is studied as in Fig. 4.10(a). When the online-adaptation SNR falls to a low SNR (i.e., 0 dB), adaptation proves ineffective, and the adapted model performs even much worse than the pretrained model. This degradation is attributed to the higher symbol-error rate in $\hat{\mathbf{X}}$ and increased noise in $\hat{\mathbf{Y}}$, both of which negatively affect SSL-task learning. If all online data are collected at a high SNR (e.g., 20 dB), the adapted model exhibits poor performance in

20011372

Table 4.1: Comparison of model parameter counts (in millions).

Model	Total params(M)	Online-trainable params(M)
HA03	**0.147**	0.147
ChannelMAE	0.206	**0.126**
DnCNN	0.228	0.227
HA02	0.272	N/A
ChannelNet	0.686	N/A



(a) Offline 3GPP channel

(b) Offline RT channel

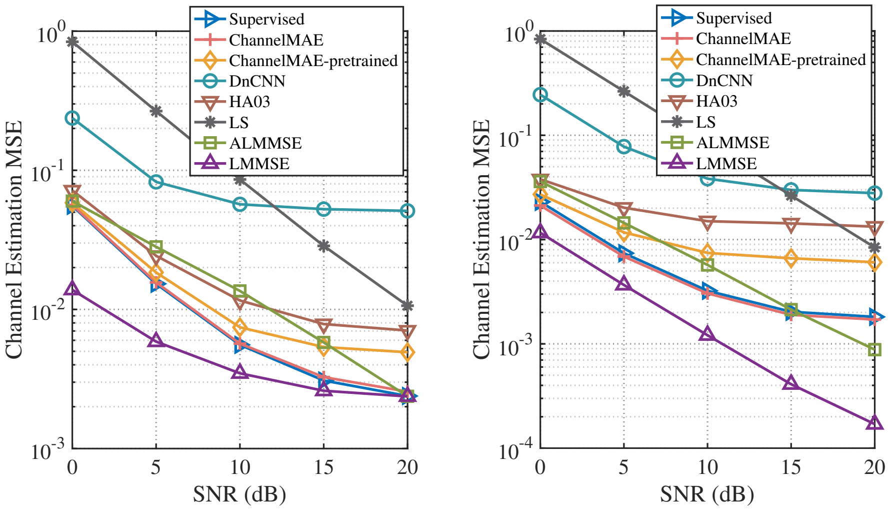
Figure 4.12: Evaluation of offline-pretrained models and conventional baselines in offline channel environments.

low-SNR regions while attaining the best MSE in high-SNR regions. Therefore, it is advantageous to include a range of online-adaptation SNRs above 5 dB. Online data augmentation is also studied shown in Fig. 4.10(b). Compared with the no-augmentation case (A=1), the five-fold augmentation scheme (A=5) achieves the overall largest performance gain, drastically outperforming the 7-fold one (A=7) at 20 dB. Thus, we adopt A=5.

Online Model Updating

Two update schemes are compared: updating the full SSL branch versus updating only the shared encoder. As shown in Fig. 4.11, the MSE gain of the adapted model over the pretrained one is computed, and the encoder-only update scheme results in a higher MSE gain compared with the full-branch update scheme, which justifies the design of adapting the shared encoder online. Meanwhile, the encoder contains 0.126M parameters while the full SSL branch has 0.153M parameters, and thus the encoder-only update scheme reduces model updating cost by roughly 18%.

20011372



(a) Online 3GPP channel (b) Online RT channel

Figure 4.13: Evaluation of online-adapted models and conventional baselines in online channel environments.

4.4.4 Comparison with Baselines

Table 4.1 compares the parameter counts of each model. Memory consumption scales with parameter count under a given floating-point quantization. Among the three architectures that support online adaptation, HA03 has the smallest overall footprint (0.147M parameters), whereas ChannelMAE requires the fewest trainable parameters during adaptation (0.126M). In the following, we elaborate the performance of both offline-pretrained and online-adapted models in 3GPP-aligned and RT channel scenarios.

Offline-Pretrained Model Performance

In offline 3GPP channels shown in Fig. 4.12(a), ChannelMAE reduces the channel-estimation MSE by 15.2–29.0% compared with HA02 across the whole SNR range. Also, its MSE is up to 95.1% lower than LS and up to 59.5% lower than ALMMSE. Although ChannelNet attains a slightly lower MSE at high SNR values of 15–20 dB than ChannelMAE, its performance significantly drops at 0 dB SNR. In offline RT channels shown in Fig. 4.12(b), ChannelMAE nearly matches the performance of LMMSE and has a lower MSE than HA02 by 6.52–47.9%. Despite ChannelNet yields comparable results to ChannelMAE, its model size is over three times larger than ChannelMAE. These results demonstrate that ChannelMAE achieves the state-of-the-art offline channel-estimation performance, while maintaining a smaller model size than its competing counterpart.

20011372

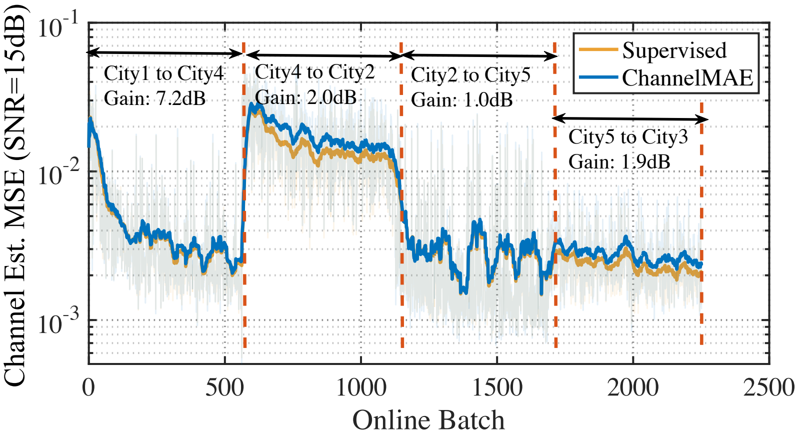
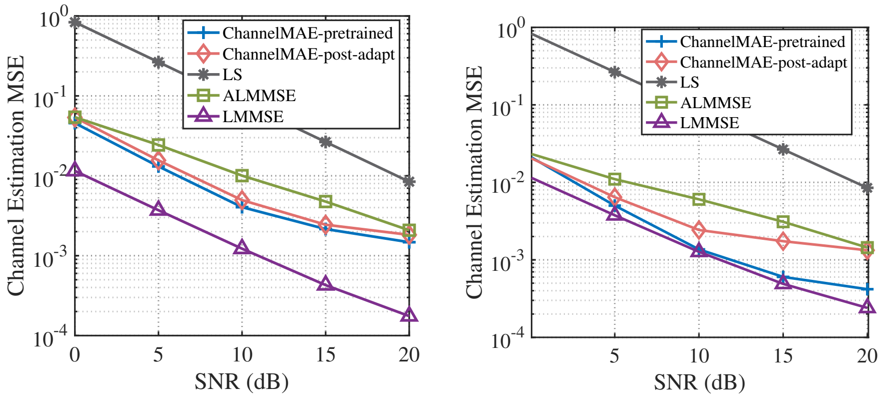


Figure 4.14: Continual online adaptation in multiple channel environment changes.



(a) Offline 3GPP channel (b) Offline RT channel

Figure 4.15: Re-evaluation of online-adapted ChannelMAE in offline channels.

Online-Adapted Model Performance

For online adaptation, models pretrained in offline 3GPP channels undergo adaptation in online 3GPP channels, while those pretrained in offline RT channels are adapted in online RT channels. These adapted models are evaluated online as seen in Fig. 4.13. The online-adapted ChannelMAE significantly achieves lower MSEs by 3.55–47.9% and 21.7–71.8% than the pretrained one in 3GPP and RT channels, respectively. In both channel environments, it reaches close to the performance of the supervised scheme using ground-truth channel coefficients, and even slightly outperforms the supervised scheme in RT channels thanks to online data augmentation. Compared with the state-of-the-art online model HA03, ChannelMAE reduces MSEs by 21.4–63.7% and 44.0–87.1% in 3GPP and RT channels, respectively.

Continual Adaptation and Forgetting Issues

A continual online adaptation process with four environmental shifts is studied, where ChannelMAE is pretrained in City1 and then adapted online throughout City4, City2, City5, and City3 sequentially. As shown in Fig. 4.14, for the first two shifts, ChannelMAE witnesses significant MSE gains evaluated at 15 dB SNR, while the model remains a low MSE for the last two shifts as channel knowledge accumulates. Throughout the adaptation process, ChannelMAE reaches close to the supervised-learning scheme. However, as ChannelMAE does not incorporate continual learning techniques to retain long-term model knowledge, forgetting issues cannot be avoided.

As depicted in Fig. 4.15, we evaluate the severity of forgetting issues. The online-adapted ChannelMAE is re-evaluated in offline environments and compared with its pretrained model. Specifically, the forgetting issue is ignorable in the 3GPP channels as shown in Fig. 4.15. However, the adapted model exhibits a noticeable performance degradation compared with the pretrained ChannelMAE in the RT channels, particularly at higher SNR values. Despite this, the adapted model still surpasses LS and ALMMSE over the whole SNR range. Future research is required to further mitigate forgetting.

4.5 Summary

To enable label-free online adaptation of pretrained neural channel estimators, an SSL task was designed on top of the original channel-estimation task. It effectively reconstructed the masked parts of randomly-masked received frames, by leveraging estimated data-symbols recovered from the online symbol-recovery mechanism. The SSL and channel-estimation tasks were then consolidated into a two-branch MAE model named ChannelMAE, where each branch was dedicated to one task and two branches shared the same encoder but used separate decoders. By online adapting this shared encoder through optimizing the SSL-task branch, the online channel-estimation performance was improved. Batch-wise online learning and online data augmentation techniques were further incorporated to reduce memory footprint in terms of data storage. ChannelMAE achieved significant performance gains over the baselines. It is the first approach that realizes online adaptation of neural channel estimators without ground-truth channel coefficients, prior channel statistics, or additional pilot overhead. The proposed two-task framework shows great promise for label-free adaptation across a broad range of wireless applications.

20011372

20011372

Chapter 5

Online Collaborative Learning among Multiple Cells for Uplink Neural Receivers

A pre-trained neural receiver does not perform well in all channel environments, so online retraining is necessary. To acquire channel knowledge efficiently, collaborative learning among multiple neural receivers is indispensable. To this end, a graph-based collaborative learning scheme called GraphRx is developed to retrain uplink neural receivers collaboratively among base stations (BSs). First, considering a collaboration graph among BSs, GraphRx is formulated as a personalized federated learning problem, wherein the graph weights and neural receiver models are learned together so that generalization and personalization are jointly optimized. Second, the problem is solved through an alternating approach under the federated learning paradigm. Particularly, an approximate generalization bound is derived to enable graph optimization at the server without accessing local data on BSs. To reduce overhead of training pilots, data augmentation is employed. GraphRx is evaluated via extensive simulation. Key parameters of GraphRx are first found through ablation study. Next, the effectiveness of the approximation in the generation bound is validated. Comparisons with the state-of-the-art schemes are finally conducted. Results show that, given the same coded bit error rate, GraphRx achieves a SNR gain of $0.4 \sim 0.9$ dB and $0.5 \sim 2.1$ dB for the cases without and with inter-cell interference, respectively.

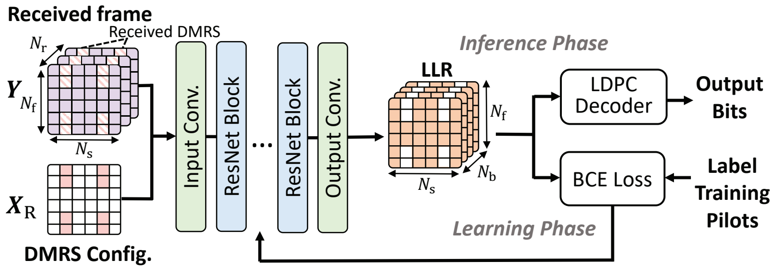


Figure 5.1: Fully convolutional network architecture of neural receiver.

5.1 System Model

5.1.1 SIMO-OFDM Communications

The uplink SIMO OFDM communication is considered with N_r receive antennas at the BS and one transmit antenna at the user. One OFDM frame spans a transmission time interval (TTI) and includes N_s OFDM symbols and N_f subcarriers. Each resource element (RE) has one symbol time and one subcarrier. Let $\mathbf{Y}, \mathbf{H}, \mathbf{N}, \mathbf{G} \in \mathbb{C}^{N_\mathrm{r} \times N_\mathrm{s} \times N_\mathrm{f}}$ be the received signals, the channel coefficients, the additive white Gaussian noise, and the inter-cell interference of one OFDM frame, respectively. The transmit signal matrix is $\mathbf{X} \in \mathbb{C}^{N_\mathrm{s} \times N_\mathrm{f}}$. Let k be the index of OFDM symbol and l be the index of subcarriers. We define the following vectors $\mathbf{y}_{kl} = \mathbf{Y}[:, k, l], \mathbf{h}_{kl} = \mathbf{H}[:, k, l], \mathbf{n}_{kl} = \mathbf{N}[:, k, l], \mathbf{g}_{kl} = \mathbf{G}[:, k, l]$ and they all have the dimension of N_r. The frequency-domain received signal on each RE during one TTI is

$$\mathbf{y}_{kl} = \mathbf{h}_{kl} x_{kl} + \mathbf{g}_{kl} + \mathbf{n}_{kl}, \tag{5.1}$$

where $x_{kl} \in \mathbf{X}, \forall k = 1, ..., N_\mathrm{s}, j = 1, ..., N_\mathrm{f}$.

5.1.2 Design of Neural Receiver

As shown in Fig. 5.1, for the model architecture of the neural receiver, a fully convolutional neural network (CNN) consisting of multiple preactivation ResNet blocks (Honkala et al., 2021) is employed. The input of neural receiver has two parts: received OFDM frame $\mathbf{Y} \in \mathbb{C}^{N_\mathrm{r} \times N_\mathrm{s} \times N_\mathrm{f}}$ and transmitted demodulation reference symbols (DMRS) configuration matrix $\mathbf{X}_\mathrm{R} \in \mathbb{C}^{N_\mathrm{s} \times N_\mathrm{f}}$. Aligned with standards (3GPP TS 38.211, 2024), DMRS occupies selected subcarriers in one or two symbol times within each TTI. To match the dimension of $N_\mathrm{s} \times N_\mathrm{f}$, zeros are inserted in non-DMRS positions within the frame to form $\mathbf{X}_\mathrm{R}$. By stacking $\mathbf{Y}$ and $\mathbf{X}_\mathrm{R}$, we have $\mathbf{T} \in \mathbb{R}^{(2N_\mathrm{r}+2) \times N_\mathrm{s} \times N_\mathrm{f}}$ by treating their real and imaginary parts as two separate channels. The output $h_{\boldsymbol{\theta}}(\mathbf{T}) \in \mathbb{R}^{N_\mathrm{s} \times N_\mathrm{f} \times N_\mathrm{b}}$ is the log-likelihood ratios (LLR) with N_RS symbols in DMRS positions ignored, where N_RS is the number of DMRS symbols within one TTI and N_b is the number of bits per symbol.

20011372

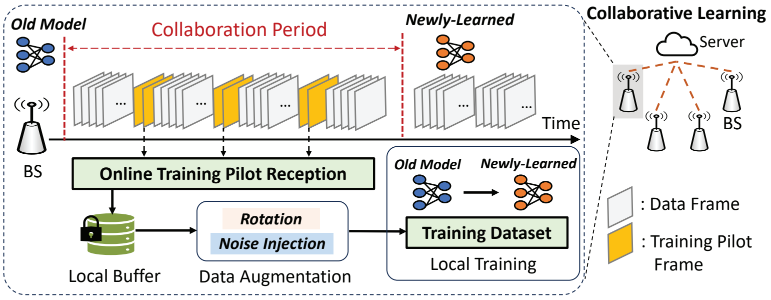


Figure 5.2: Working process of each BS in collaborative learning.

For model inference, $\mathbf{Y}$ contains received uplink data except REs in DMRS positions. The model output (i.e., LLR of data bits) is directly fed into an LDPC channel decoder to obtain detected bits. For model training, $\mathbf{Y}$ contains *received training pilots* that are transmitted in place of uplink data. Since bit detection is a binary classification problem for each bit, the model output $h_{\boldsymbol{\theta}}(\mathbf{T})$ is compared against label $\mathbf{L}$ to compute the binary cross-entropy (BCE) loss after Sigmoid activation (Honkala et al., 2021), where $\mathbf{L}$ is the preknown training pilot bits of length $(N_{\mathrm{s}}N_{\mathrm{f}} - N_{\mathrm{RS}})N_{\mathrm{b}}$. In this chapter, transmission and reception of training pilots specially serve for model retraining, while system-configured DMRS are kept for a neural receiver to implicitly perform channel estimation by using $\mathbf{X}_{\mathrm{R}}$ and received DMRS symbols in the corresponding positions of $\mathbf{Y}$ (Honkala et al., 2021).

5.1.3 Elements of Supervised Learning Problem

There are M base stations and one central server in a multi-cell network shown in Fig. 5.2. The set of BS indices is denoted by $\mathcal{M} = \{1, ..., M\}$. An OFDM frame carrying training pilots is named *training pilot frame* while a normal frame carrying communication data is named *data frame*. BS m builds up an online dataset $\mathcal{Z}_m$ locally via periodic reception of training pilot frames (Honkala et al., 2021). Local dataset $\mathcal{Z}_m = \{(\mathbf{T}_m^{(i)}, \mathbf{L}_m^{(i)})\}_{i=1}^{N_m}$ has N_m data instances, where $\mathbf{T}_m^{(i)}$ and $\mathbf{L}_m^{(i)}$ are the input and label of the i-th data instance. Let $\mathbf{p} = [p_1, ..., p_M]$ be the data quantity vector, i.e., $p_m = \frac{N_m}{N}$, where $N = \sum_{m=1}^{M} N_m$ denotes the total number of data instances. Empirical risk function $\hat{F}(\cdot)$ over local dataset $\mathcal{Z}_m$ is defined as

$$\hat{F}(\boldsymbol{\theta}; \mathcal{Z}_m) = \frac{1}{N_m} \sum_{i=1}^{N_m} \ell\left(h_{\boldsymbol{\theta}}\left(\mathbf{T}_m^{(i)}\right), \mathbf{L}_m^{(i)}\right), \tag{5.2}$$

where $h_{\boldsymbol{\theta}}$ is a mapping function parameterized by model parameters $\boldsymbol{\theta}$, and $\ell(\cdot)$ is the BCE loss stated previously. In this chapter, each BS has a personalized neural receiver denoted by $\boldsymbol{\theta}_m, \forall m \in \mathcal{M}$.

5.2 Problem Formulation and Analysis

The overall working procedure of each BS is illustrated in Fig. 5.2. During online deployment, each BS collects a local dataset via periodic reception of training pilot frames within a fixed time interval named *collaboration period*. To avoid disrupting normal communications, training pilot frames are sparsely inserted among data frames, such as at a ratio of one training pilot frame for every 1000 data frames. This local dataset is further augmented to enrich its quantity and diversity, as elaborated in Section 5.4.3.

Based on the collected data, each BS participates in multi-cell collaborative learning coordinated by the cloud server, such that it learns data distributions from other cells. It is emphasized that collaborative learning is a large-timescale task (Polese et al., 2023) running in the background, as sufficient online data must be collected for a neural receiver to learn statistical features of online channels effectively. In addition, collaborative learning will not disrupt online inference of the currently deployed neural receiver. Once a new neural receiver is learned via collaborative learning, it replaces the old one in operation. Local real-time adaptation of neural receiver (Fischer et al., 2022) may be incorporated in the framework to temporarily enhance performance, but it is not the focus of this chapter. Next, the graph-based collaborative learning problem is formulated.

5.2.1 Collaboration-Graph-Based Learning

To reveal the pair-wise collaboration formed among BSs, a directed collaboration graph weights $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is defined: node set $\mathcal{V} = \{\boldsymbol{\theta}_m\}_{m\in\mathcal{M}}$ is the set of local models, and edge set $\mathcal{E} = \{(j, m)|w_{mj} \geq 0, \forall m, j \in \mathcal{M}\}$, where w_{mj} is the edge weight on the directed edge from node j to node m. This edge weight also indicates on what level node m should learn from the data distribution of node j. There is no collaboration between two nodes if $w_{mj} = 0$. Let $\mathbf{W}$ be the directed graph weight matrix with w_{mj} as its element. Let $\mathbf{w}_m = [w_{m1}, ...w_{mM}]$ be the collaboration vector of BS m, and $\mathcal{N}_m = \{j|j \neq m, w_{mj} > 0, j \in \mathcal{M}\}$ be the neighbor set of BS m.

Collaboration graph weights $\mathbf{W}$ is unknown and thus must to be optimized along with personalized model parameters $\{\boldsymbol{\theta}_j\}_{j\in\mathcal{M}}$. The existing graph-based PFL formulation (Ye et al., 2023) embeds the graph weights into both the aggregated model term and the graph regularization term, which has two problems: 1) the second term is redundant when the first term already exists; 2) the second term based on cosine similarity only measures the angular difference between model vectors but ignores their magnitude difference.

To this end, a new formulation for graph-based learning is designed to explicitly capture the data-level collaboration from the perspective of each BS: BS m trains its personalized model $\boldsymbol{\theta}_m$ by learning from local data $\mathcal{Z}_m$ and diverse data from its neighbors $\mathcal{Z}_j, j \in \mathcal{N}_m$, such that the learned personalized model enjoys better generalization ability; meanwhile, collaboration weight w_{mj} controls how much diverse data distributions

20011372

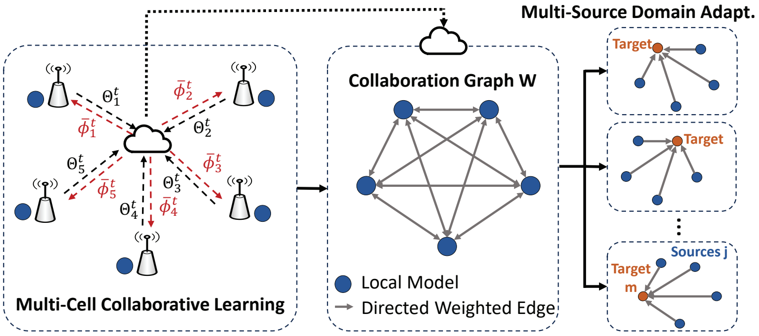


Figure 5.3: GraphRx: graph-based collaborative learning scheme.

contribute to the model learning such that collaboration will not cause local performance degradation.

Based on the above design rational, the graph-based collaborative learning of personalized model $\boldsymbol{\theta}_m, \forall m \in \mathcal{M}$ is formulated as Problem 1 (P1):

$$\boldsymbol{\theta}_m, \mathbf{w}_m = \arg \min_{\boldsymbol{\theta}, \mathbf{w}_m} \sum_{j \in \{m\} \cup \mathcal{N}_m} w_{mj} \hat{F}\left(\boldsymbol{\theta}; \mathcal{Z}_j\right) \tag{5.3}$$

$$s.t. \quad \sum_{j=1}^{M} w_{mj} = 1, w_{mj} \geq 0, \forall j \in \mathcal{M}, \tag{5.4}$$

where the sum of non-negative weights in a collaboration vector is normalized to 1 in (5.4). The collaboration graph can flexibly balance the contribution of local knowledge and collaborative knowledge to capture the personalization-generalization tradeoff. If the collaborative-knowledge term denoted by $\sum_{j \in \mathcal{N}_m} w_{mj} \hat{F}(\boldsymbol{\theta}; \mathcal{Z}_j)$ is excluded from the objective (5.3), P1 degenerates to a local learning problem without inter-BS collaboration. On the other hand, if the collaboration graph $\mathbf{W}$ is set to be fully-connected with $\mathbf{w}_m = p_m \vec{\mathbf{1}}$ and there exists only one universal model, P1 falls to a conventional FL problem (McMahan et al., 2017).

5.2.2 Problem Analysis

Assuming a centralized setting where all the data and models are collected to the server, the server can directly solve P1 for each BS in parallel. However, the FL setting in our problem imposes two critical constraints (McMahan et al., 2017): 1) only models are communicated between the server and BSs; 2) there is no peer-to-peer communication between BSs. These constraints render solving P1 for each BS much more challenging. Following the principle of optimization decomposition, P1 is decomposed into two alternative steps: 1) at each BS m, optimizing local model $\boldsymbol{\theta}_m$ with collaboration graph $\mathbf{W}$ fixed; 2) at the server, optimizing collaboration graph $\mathbf{W}$ given all the collected local models $\{\boldsymbol{\theta}_j\}_{j \in \mathcal{M}}$.

For the first step, to fulfill collaboration while preserving data locality, local model update is combined with weighted model aggregation. Note that model aggregation is performed based on $\mathbf{W}$ at the server side. As shown in Fig. 5.3, each BS m first fetches an aggregated model denoted as $\bar{\boldsymbol{\phi}}_m$ from the server as local model initialization. Each BS then runs local gradient descent to update its local model. The details will be elaborated in Section 5.4.1.

For the second step, however, it stills remains a critical challenge to determine the collaboration graph. The server can only access models $\{\boldsymbol{\theta}_m\}_{m\in\mathcal{M}}$ while it cannot access local data to evaluate empirical risks $\hat{F}(\cdot)$. Therefore, objective (5.3) cannot be optimized directly. In the following section, a new methodology derived from the generalization bound theory is designed.

5.3 Collaboration Graph Optimization

To optimize the collaboration graph, the theoretical generalization bound of $\{\boldsymbol{\theta}_m\}_{m\in\mathcal{M}}$ is first derived. It is expressed by a function of collaboration graph weights $\mathbf{W}$. Next, the bound is approximated empirically given the current personalized models, and the approximate bound is used as an empirical objective for optimizing graph weights $\mathbf{W}$.

5.3.1 Derivation of Generalization Bound

In the rest of the chapter, let $\hat{F}_m(h) = \hat{F}(h; \mathcal{Z}_m)$ for ease of notation. In preparation for theoretical derivation, let a hypothesis function, denoted by $h \in \mathcal{H}$, be a general mapping from the model's input to its output, where $\mathcal{H}$ is the hypothesis space quantified by finite Vapnik-Chervonenkis (VC) dimension c Ben-David et al. (2010).

Let $F_m(h)$ be the *expected risk* of a hypothesis h over true data distribution $\mathcal{D}_m$ at BS m, which is

$$F_m(h) = \mathbb{E}_{\mathbf{T}\sim\mathcal{D}_m}[\ell(h(\mathbf{T}), \mathbf{L})], \tag{5.5}$$

where $\mathbf{T}$ is sampled from $\mathcal{D}_m$.

The fundamental goal of collaborative learning is to improve the generalization ability of each hypothesis to unseen data instances, which is equivalent to minimizing the expected risk denoted by $F_m(\cdot)$. However, in practice, true data distribution $\mathcal{D}_m$ cannot be known. Therefore, each node has to minimize the empirical risk (i.e., objective (5.3)) over limited data instances from a variety of data sources. Given any collaboration vector $\mathbf{w}_m$ that satisfies constraint (5.4), the objective of each BS m in (5.3) is expressed as minimizing the empirical $\mathbf{w}_m$-weighted risk $\hat{F}_{\mathbf{w}_m}(h)$:

$$\hat{h}_{\mathbf{w}_m} = \arg\min_{h\in\mathcal{H}} \hat{F}_{\mathbf{w}_m}(h), \tag{5.6}$$

20011372

where $\hat{F}_{\mathbf{w}_m}(h) = \sum_{j=1}^{M} w_{mj}\hat{F}_j(h)$ and $\hat{h}_{\mathbf{w}_m}$ is the optimal minimizer. Note that the corresponding expected $\mathbf{w}_m$-weighted risk is $F_{\mathbf{w}_m}(h) = \sum_{j=1}^{M} w_{mj}F_j(h)$.

To evaluate how $\hat{h}_{\mathbf{w}_m}$ generalizes over the true data distribution, the generalization bound of $\hat{h}_{\mathbf{w}_m}$ is expressed as the upper bound of the expected risk, i.e., $F_m(\hat{h}_{\mathbf{w}_m})$. This generalization bound is derived in the following theorem.

Theorem 5.1. *$\mathcal{H}$ is a hypothesis space of VC dimension c. For each $j \in \mathcal{M}$, $N \cdot p_j$ data points are drawn from distribution $\mathcal{D}_j$. For any $\delta \in (0,1)$, with probability at least $1-\delta$, there exists*

$$F_m(\hat{h}_{\mathbf{w}_m}) \leq F_m(h_m^*) + 2\sqrt{\sum_{j=1}^{M} \frac{w_{mj}^2}{p_j}\left(\frac{c\log(2N) - \log\delta}{2N}\right)} + \sum_{j=1}^{M} w_{mj}\left(d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_m, \mathcal{D}_j) + 2\lambda_{mj}\right),$$

where $h_m^* = \arg\min_{h\in\mathcal{H}} F_m(h)$ is the optimal minimizer of expected risk, and $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_m, \mathcal{D}_j)$ is the $\mathcal{H}\Delta\mathcal{H}$-divergence (Ben-David et al., 2010), and $\lambda_{mj} = \min_{h\in\mathcal{H}}(F_m(h) + F_j(h))$.

The proof is given in Appendix C.

For ease of notation, let $d_{mj} = d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_m, \mathcal{D}_j) + 2\lambda_{mj}$, and $\mathbf{D}$ is defined as the distribution divergence matrix whose element at the m-th row and the j-th column is pair-wise divergence d_{mj}. Inspired by multi-source DA, for each node on the collaboration graph, it can be considered in a target domain with its neighbor nodes in source domains. Thus, a graph-based learning problem can be decomposed into the parallel multi-source DA processes shown in Fig. 5.3. The generalization bound in Theorem 1 is derived for each BS m, and thus the total generalization bound is obtained via the summation of all the upper bounds:

$$\sum_{m=1}^{M} F_m(\hat{h}_{\mathbf{w}_m}) \leq \sum_{m=1}^{M} F_m(h_m^*) + 2Q(\mathbf{W}, \mathbf{p}, N, c) + \sum_{m,j\in\mathcal{M}} [\mathbf{W} \odot \mathbf{D}]_{mj}, \tag{5.7}$$

where

$$Q(\mathbf{W}, \mathbf{p}) = \sum_{m=1}^{M} \sqrt{\sum_{j=1}^{M} \frac{w_{mj}^2}{p_j}\left(\frac{c\log(2N) - \log\delta}{2N}\right)}, \tag{5.8}$$

$$\sum_{m,j\in\mathcal{M}} [\mathbf{W} \odot \mathbf{D}]_{mj} = \sum_{m=1}^{M}\sum_{j=1}^{M} w_{mj}d_{mj}. \tag{5.9}$$

The total generalization bound shown in Eq. (5.7) contains three terms: 1) the first term, $\sum_{m=1}^{M} F_m(h_m^*)$, is deemed as a constant value irrelevant to the graph weights; 2) the second term, $Q(\mathbf{W}, \mathbf{p}, N, c)$, is a *quantity-aware term* that depends on data quantity vector $\mathbf{p}$, graph weights $\mathbf{W}$, total number of data instances N, and VC dimension c of the hypothesis space; 3) the third term, $\sum_{m,j\in\mathcal{M}}\left[\mathbf{W}\circ\mathbf{D}\right]_{mj}$, is the *distribution-divergence-aware term* depending on collaboration weights $\mathbf{W}$ and distribution divergence matrix $\mathbf{D}$.

When the theoretical generalization bound (right-hand-side of Eq. (5.7)) is minimized, the total expected risk of all the hypothesis functions has the lowest upper bound, leading to the best generalization ability. Therefore, the fundamental insight is to find a collaboration graph that minimizes the generalization bound in Eq. (5.7), and thus the obtained graph can achieve the best generalization ability for personalized models. Specifically, within the bound, the quantity-aware term and the distribution-divergence-aware term (both are dependent on $\mathbf{W}$) need to be minimized. However, due to the difficulty of computing these two theoretical terms without data, they must be approximated such that an empirical objective of bound minimization can be formed at the server side.

5.3.2 Empirical Objective for Optimizing Collaboration Graph

To find an empirical objective for optimizing the collaboration graph, three steps are taken. First, the quantity-aware term in Eq. (5.8) is approximated. Complexity indicator is defined as $C = 2\sqrt{(c\log(2N) - \log\delta)/2N}$. It is treated as a hyper-parameter embedded in the quantity-aware term.

Second, the distribution-divergence-aware term in Eq. (5.9) is approximated. Pair-wise divergence d_{mj} is approximated by the distance between the two models, $\hat{d}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_j)$. Note that this approximation of distribution divergence is adaptively adjusted as the model parameters are iteratively updated. The positive correlations between the distribution divergence and the model difference are observed in Ye et al. (2023); Zantedeschi et al. (2020), and it is also proved in Zhao et al. (2018) that the statistical distance between two distributions directly causes model parameter difference. In Section 5.5.3, this approximation is validated by comparing model difference with estimated distribution divergence.

Third, combining the above approximate terms, an empirical optimization objective regarding $\mathbf{W}$ is formed as

$$\hat{B}(\mathbf{W}, \boldsymbol{\Theta}) = C\sum_{m=1}^{M}\sqrt{\sum_{j=1}^{M}\frac{w_{mj}^2}{p_j}} + \sum_{m=1}^{M}\sum_{j=1}^{M} w_{mj}\hat{d}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_j). \tag{5.10}$$

By minimizing (5.10), the optimal collaboration graph is obtained, leading to the lowest approximate generalization bound.

20011372

5.4 Graph-Based Collaborative Learning

Two alternating steps stated in Section 5.2 are elaborated. In addition, the data augmentation mechanism is adopted to enrich training data.

5.4.1 Alternating Optimization

The designed collaborative learning process consists of the alternation of two steps: 1) at each BS m, optimizing the local model $\boldsymbol{\theta}_m$ with the collaboration graph $\mathbf{W}$ fixed; 2) at the server, optimizing the collaboration graph $\mathbf{W}$ given all the local models $\{\boldsymbol{\theta}_m\}_{m\in\mathcal{M}}$. Let t denote the index of communication round.

Optimizing Local Model at Each BS

The distributed model updating process follows a widely-used protocol in distributed learning. Each BS first fetches the aggregated model denoted by $\bar{\boldsymbol{\phi}}_m^t$ from the server. Each BS then uses it to initialize the local model as

$$\boldsymbol{\theta}_m^{t,0} \leftarrow \bar{\boldsymbol{\phi}}_m^t, \tag{5.11}$$

where $\boldsymbol{\theta}_m^{t,0}$ is the initial model. Suppose $\boldsymbol{\theta}_m^{t,i}$ is the model after the i-th mini-batch gradient descent, then it is updated by minimizing local empirical risk $\hat{F}_m(\cdot)$ in the next step as

$$\boldsymbol{\theta}_m^{t,i+1} \leftarrow \boldsymbol{\theta}_m^{t,i} - \eta\nabla\hat{F}_m(\boldsymbol{\theta}_m^{t,i}), \tag{5.12}$$

with η as the learning rate. After s steps starting from $\boldsymbol{\theta}_m^{t,0}$, each BS sends the updated local model $\boldsymbol{\theta}_m^{t,s}$ to the server.

Optimizing Collaboration Graph at the Server

As stated in Section 5.3, the empirical objective (5.10) guided by the generalization bound is minimized:

$$\begin{aligned} \mathbf{W}^t &= \arg\min_{\mathbf{W}} \hat{B}(\mathbf{W}, \boldsymbol{\Theta}) && (5.13)\\ s.t. \quad & \sum_{j=1}^{M} w_{mj} = 1, w_{mj} \geq 0, \forall j, m \in \mathcal{M}. && (5.14) \end{aligned}$$

The above problem is a convex optimization, and thus its global optimum is solved by conventional solvers (Boyd and Vandenberghe, 2004). Once the current collaboration graph $\mathbf{W}^t$ for $\{\boldsymbol{\theta}_m^{t,s}\}_{m\in\mathcal{M}}$ is obtained, the

Algorithm 4 GraphRx

```
Inputs: Total number of rounds $T$, initial local models $\boldsymbol{\theta}_m^0 = \psi, \forall m \in \mathcal{M}$, collaboration graph weights $\mathbf{W}^0$, $\bar{\boldsymbol{\phi}}_m^0 = \mathbf{0}$, number of steps $s$.
for each communication round $t = 1, \ldots, T$ do
    /*Optimize local models at BS side:*/
    for each BS $m \in \mathcal{M}$ in parallel do
        Initialize the local model by Eq. (5.11);
        Run $s$ steps of local gradient descent by Eq. (5.12);
        Send the updated local model $\boldsymbol{\theta}_m^{t,s}$ to the server;
    end for
    /*Optimize collaboration graph at server side:*/
    The server obtains $\mathbf{W}^t$ by solving Eq. (5.13-5.14);
    The server aggregates models for each BS $m$: $\bar{\boldsymbol{\phi}}_m^{t+1} = \sum_{j \in \{m\} \cup \mathcal{N}_m} w_{mj}^t \boldsymbol{\theta}_j^{t,s}, \forall m \in \mathcal{M}$;
    The server sends the aggregated model $\bar{\boldsymbol{\phi}}_m^{t+1}$ to each BS $m$.
end for
Outputs: $\{\boldsymbol{\theta}_m^T\}_{m \in \mathcal{M}}, \mathbf{W}^T$.
```

server computes the aggregated model for each BS m, which is expressed by

$$\bar{\boldsymbol{\phi}}_m^{t+1} = \sum_{j \in \{m\} \cup \mathcal{N}_m} w_{mj}^t \boldsymbol{\theta}_j^{t,s}, \forall m \in \mathcal{M}, \tag{5.15}$$

which will be sent to each BS m in the next round.

Based on the above two steps, such T_{c} communication rounds of training are performed until each BS m obtains a well-learned personalized neural receiver $\boldsymbol{\theta}_m^T$ with multi-cell collaborative knowledge and the learned collaboration graph is $\mathbf{W}^T$.

5.4.2 Algorithm Design

The designed PFL algorithm GraphRx is stated in Algorithm 4.The shared pretrained neural receiver ψ is deployed at each BS as the initial model, i.e., $\boldsymbol{\theta}_m^0 = \psi, \forall m \in \mathcal{M}$. The collaboration graph weights $\mathbf{W}^0$ is initialized as a fully-connected graph with $\mathbf{w}_m = p_m \vec{\mathbf{1}}$. At the beginning of each communication round t, for each BS, it first initializes its local model using the aggregated model as in Eq. (5.11), and then performs s steps of local mini-batch gradient descent. Each BS sends the local model to the server. At the server side, with all the local models collected, the server solves the optimal collaboration graph weights $\mathbf{W}$. Based on the obtained $\mathbf{W}$, the server performs weighted model aggregation for each BS. the server sends the aggregated model $\bar{\boldsymbol{\phi}}_m^{t-1}$ to each BS m. Such T communication rounds of training are performed until each BS m obtains a well-learned personalized neural receiver $\boldsymbol{\theta}_m^T$ with multi-cell collaborative knowledge.

5.4.3 Mechanism of Data Augmentation

To achieve high spectral efficiency, each BS must keep the overhead of training pilots low. Thus, the overall quantity of training pilots may not be sufficient to produce satisfying retraining results within a relatively short collaboration period. To this end, two data augmentation techniques, rotation and noise injection, are employed at each BS to enrich the quantity and diversity of online training data.

Specifically, we adopt the design in Fischer et al. (2022): 1) a rotating angle ϕ is uniformly sampled from $[0, 2\pi)$, and each symbol in received training pilot frame $\mathbf{Y}$ is applied such a rotation to obtain $e^{j\phi}\mathbf{Y}$; 2) an additive noise $\mathbf{N}_{\mathrm{a}}$ (with the same dimension as $\mathbf{N}$) is injected to each symbol, where $\mathbf{N}_{\mathrm{a}} \sim \mathcal{CN}(0, \sigma_a^2)$. σ_a^2 is uniformly sampled from $(0, \frac{\sigma^2}{2}]$ (σ^2 is the noise variance of $\mathbf{N}$). Thus, the augmented frame (denoted as $\mathbf{Y}_{\mathrm{a}}$) is obtained: $\mathbf{Y}_{\mathrm{a}} = e^{j\phi}\mathbf{Y} + \mathbf{N}_{\mathrm{a}}$. A new instance can be created by replacing $\mathbf{Y}$ with $\mathbf{Y}_{\mathrm{a}}$ during training with the label unchanged, and $\mathbf{Y}_{\mathrm{a}}$ will not be stored. Thus, no additional storage cost is incurred. If the above augmentation is applied $A-1$ times for each original instance, the augmented training data quantity is increased by A times (i.e., augmentation time).

5.5 Performance Evaluation

In this section, the simulation set-up is first presented, followed by the results of ablation study, design verification, and comparisons with several baselines.

5.5.1 Simulation Setup

For uplink SIMO communications, we assume one antenna at the user terminal and two receive antennas at the BS. The number of BSs in multi-cell collaboration is set to 6, i.e., $M = 6$. Heterogeneous channel environments are assumed for these six cells. Specifically, the standard tapped delay line (TDL) channel models (3GPP TS 38.901, 2024) are adopted. Four diverse channel models (i.e., TDL-A/B/C/E) each with a unique power delay profile (PDP) are used. Based on the existing evaluations (Fischer et al., 2022), diverse distributions of channel coefficient matrix $\mathbf{H}$ can be constructed with different channel profiles and different values of fast-fading parameters, such as root-mean-squared (RMS) delay spread σ_{ds} and terminal velocity interval v. Regarding training data generation, the training pilots are random binary sequences and DMRS is configured at the second and eleventh symbol times within each TTI (3GPP TS 38.211, 2024). The batch size is set to 32, i.e., 32 TTIs/batch.

The offline channel distribution is specified as the TDL-A profile in the RMS delay spread interval of 0-50 ns

20011372

Table 5.1: Heterogeneous multi-cell environments

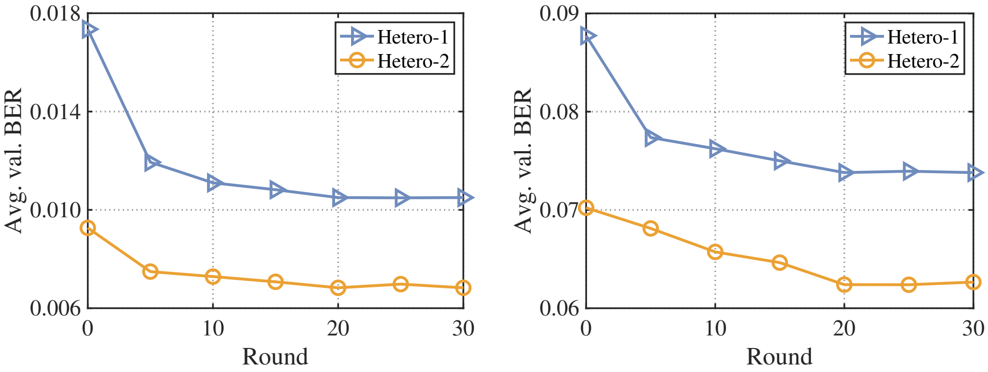
BS ID	Hetero-1			Hetero-2		
	PDP	$\sigma_{ds}(ns)$	$v(m/s)$	PDP	$\sigma_{ds}(ns)$	$v(m/s)$
1	B	500-600	0-5	B	0-50	0-5,15-20
2	B	450-550	0-5	C	450-500	0-5,15-20
3	C	200-300	15-20	B	0-50	0-5
4	C	150-250	15-20	C	150-200	0-5
5	E	400-600	15-20	E	0-50	0-5,15-20
6	E	450-550	15-20	E	200-400	0-5,15-20

and the velocity interval of 0-5 m/s over the SNR range [-4,15] dB. The neural receiver is pretrained over 2,000 batches offline and then deployed online (the initial model for GraphRx). To generate each channel instance, the exact values of t_{ds}, v, and SNR are uniformly sampled from the given intervals, and the channel coefficients are decided accordingly.

For online multi-cell channel generation, there are two types of online heterogeneous channel distributions among six cells as shown in Table 5.1, namely heterogeneous data distributions 1 and 2 (i.e., Hetero-1 and Hetero-2). In the first type, within each of these three pairs (BSs 1 and 2, BSs 3 and 4, BSs 5 and 6), two BSs exhibit similar channel distributions. Each BS holds the same training data quantity, i.e., 200 batches. In the second type, a more complicated heterogeneity is presented, and the training data quantity ratio of six BSs is 3:3:1:1:2:4. The total number of training data batches is 1,400. The online SNR range is set to [0,10] dB. During training, 20% of training data batches are used for validation, where a single-point BER (*validation BER*) is evaluated over validation data. To test the performance of a trained neural receiver, the standard Monte Carlo simulation (3GPP TS 38.211, 2024) is performed across the SNR range of [-4,15] dB, where each channel instance is randomly generated based on Table 5.1.

In addition, inter-cell interference is generated by letting another signal propagate through an interfering channel and reach the interfered BS (Honkala et al., 2021). For each uplink of BS m, the interfering PDP is randomly selected among all the other BSs' channel profiles, and RMS delay spread of interfering channel has a deviation of ±20 ns from that of uplink channel. The interference power is on average 5 dB lower than the noise power (3GPP TS 38.133, 2024). Note that the above interfering channel information is not known by the receiver.

The critical simulation parameters are as follows. In each cell, the system carrier frequency is 4 GHz with subcarrier frequency 15kHz and OFDM symbol duration 71μs, and $N_{\mathrm{f}} = 72, N_{\mathrm{s}} = 14, N_{\mathrm{RS}} = 48$. The test modulation scheme is 16-QAM and LDPC channel coding with the code rate as 658/1024 is adopted. Unless otherwise specified, the hyper-parameters of GraphRx are as follows: the number of epochs for local gradient descent set as 2; initial collaboration graph $\mathbf{W}^0 = \frac{1}{6}\vec{\mathbf{1}}$; complexity indicator $C = 0.6$, data augmentation time



(a) Without interference (b) With inter-cell interference

Figure 5.4: Average validation BER at various communication rounds.

$A = 4$, number of communication rounds $T_\mathrm{c} = 20$, which are confirmed via ablation study. The ℓ_2-norm is adopted as the model distance metric stated in Section 5.3.2, i.e., $\hat{d}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_j) = \|\boldsymbol{\theta}_m - \boldsymbol{\theta}_j\|_2$. Each distance value is scaled into $[0, 1]$. For the training of neural receiver, the Adam optimizer with the learning rate $\eta = 0.001$ is adopted.

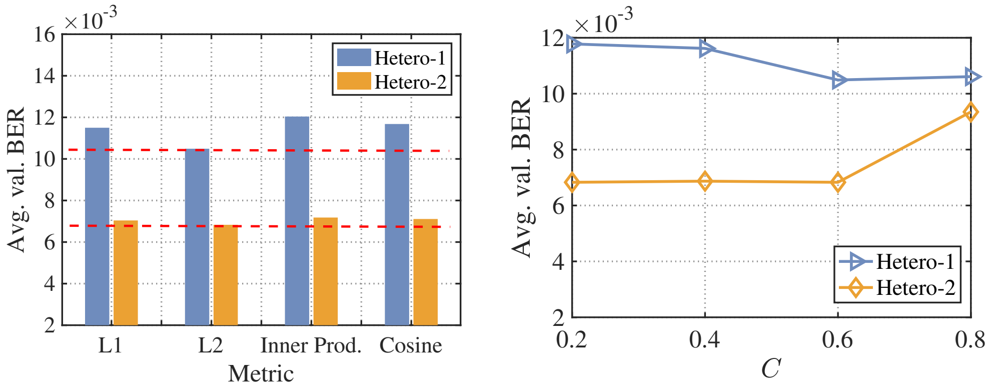
The main performance metric for evaluating receiver performance is coded bit-error rate (BER). Our method GraphRx is compared with the following baselines: 1) practical LMMSE receiver (Honkala et al., 2021); 2) the locally-trained neural receivers with no collaboration; 3) the FedAvg-trained globally-unified neural receiver (McMahan et al., 2017); 4) the neural receivers trained by the state-of-the-art PFL algorithms. Specifically, three representative PFL algorithms are selected: Ditto (Li et al., 2021), FedRep (Collins et al., 2021), and pFedGraph (Ye et al., 2023). Furthermore, the genie-aided LMMSE receiver with full and perfect channel coefficients is considered to achieve the upper-bound BER performance in the interference-free case.

5.5.2 Ablation Study

Impact of Communication Round

The number of communication rounds for GraphRx to converge is explored. During training, the validation BER is evaluated at 4 dB for each neural receiver. As shown in Fig. 5.4, in both interference-free and interference-included scenarios, GraphRx reaches convergence at the 20-th communication round, where the average validation BER of all the BSs remains stable in the following rounds. The existence of training data quantity imbalance and inter-cell interference does not affect the convergence of GraphRx.

20011372



(a) Model distance metric

(b) Complexity indicator value

Figure 5.5: Average validation BER with various distance metrics and complexity indicator values (i.e., C).

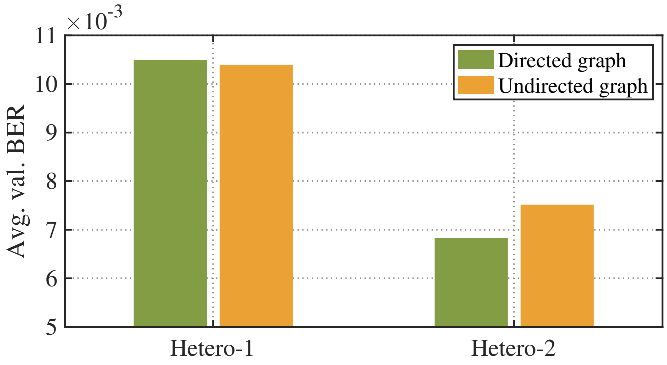


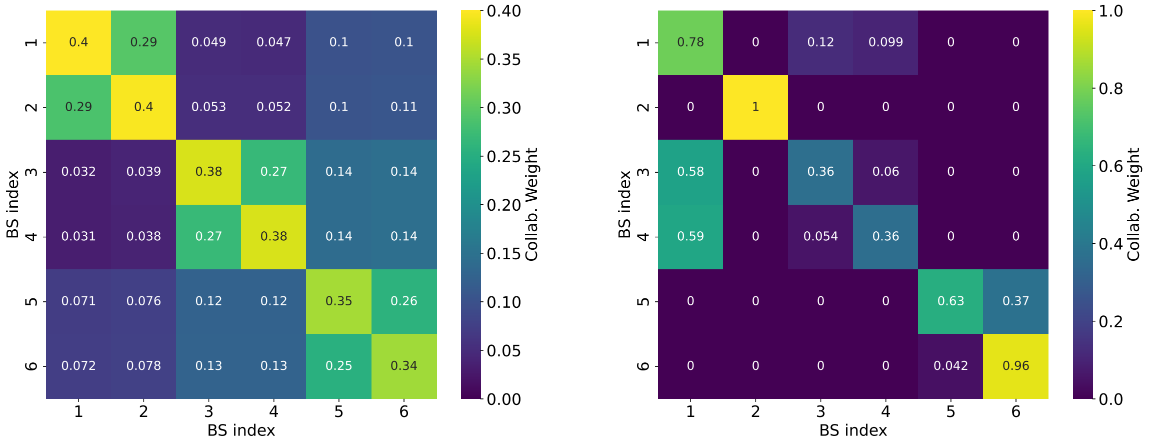
Figure 5.6: Average validation BER with directed/undirected graphs.

Impact of Model Distance Metrics and Complexity Indicator Values

As stated in Section 5.3.2, model distances $\hat{d}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_j)$ are computed to approximate distribution divergences. Four distance metrics (i.e., ℓ_1/ℓ_2-norm, inner product, and cosine similarity) are evaluated for two heterogeneous distributions, respectively. As shown in Fig. 5.5(a), ℓ_2-norm achieves the lowest validation BER under two types of heterogeneity, and thus ℓ_2-norm is selected as the desired metric. In addition, the complexity indicator C in the objective (5.10) is varied from 0.2 to 0.8, which means the importance of quantity-aware term increases. As shown in 5.5(b), for Hetero-1, C needs to be between 0.6 and 0.8, while for Hetero-2, low BERs are achieved with C varying from 0.2 to 0.8. Therefore, this complexity indicator is set to 0.6.

Impact of Collaboration Graph Directivity and Visualization of Learned Graphs

Since the collaboration relationship is not necessarily reciprocal, a directed collaboration graph is learned in GraphRx. To validate this point, the performance of GraphRx with directed and undirected graphs are compared



(a) Hetero-1

(b) Hetero-2

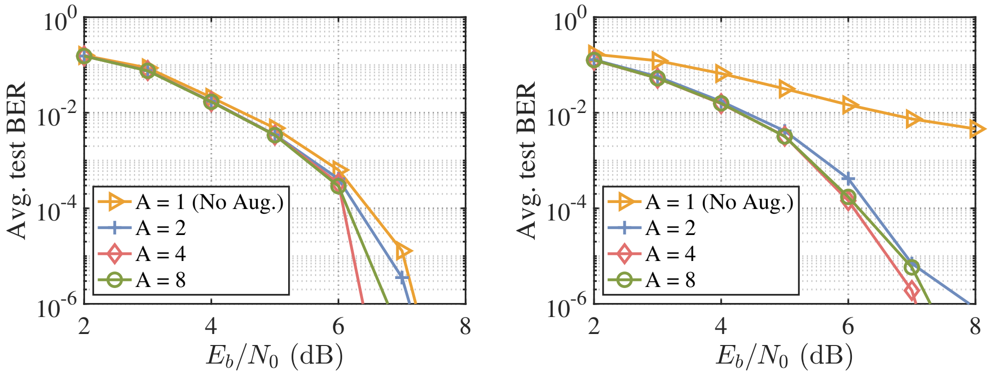
Figure 5.7: Visualization of directed collaboration graphs.

in Fig. 5.6. Note that graph weight matrix **W** must be symmetric in an undirected graph, which is added as a constraint in solving (5.13)-(5.14). For Hetero-1 where all BSs hold the same quantity of data, the graph directivity has little impact on validation BER, while the BER drops by 9.13% for Hetero-2 if a directed graph is learned instead of an undirected one. This is because for Hetero-2 each BS has various data quantities: the contribution from BSs with large data quantities to those with smaller quantities is more significant than the reverse, which is also validated by the following graph visualization.

The learned collaboration weights w_{mj} of directed graphs with GraphRx are shown in Fig. 5.7 for two heterogeneous distributions. Specifically, w_{mj} is the value at the m-th row and the j-th column, which indicates how much BS m learns form BS j. In Fig. 5.7(a), there exist three pairs of BSs (1,2), (3,4), (5,6) exhibiting strong collaboration weights. This is well aligned with the characteristics of their true data distributions shown in Table 5.1. In Fig. 5.7(b), BS 2 does not need to collaborate with others based on the learned graph, as it holds an enough quantity of data and may not benefit from learning other BSs' data. BS 5 benefits much more by learning from BS 6 than the reverse. Also, strong uni-directional collaboration is established in each BS pair of (3,1) and (4,1).

Validation of Data Augmentation Mechanism

The evaluation results of the data augmentation mechanism is presented for Hetero-1 in both interference-free and interference-included cases. Specifically, through extensive trials, a quantity of around 200 unaugmented data batches is required at each BS to guarantee convergence of GraphRx. As the training pilot overhead is



(a) Without interference (b) With inter-cell interference

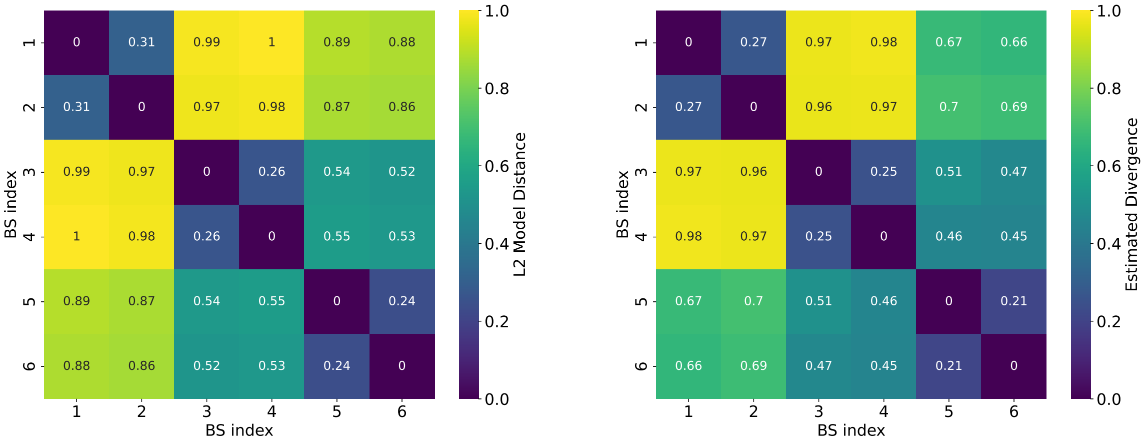
Figure 5.8: Coded BER of GraphRx tested under various augmentation times.

controlled as low as 0.1%, one training pilot frame is inserted among 1,000 transmitted OFDM frames. In this case, it requires around 1-2 hours (collaboration period) for each BS to collect 200 unaugmented batches (6,400 TTIs). As previously stated in Sec. 5.2, during the collaboration period, the currently deployed neural receiver operates normally.

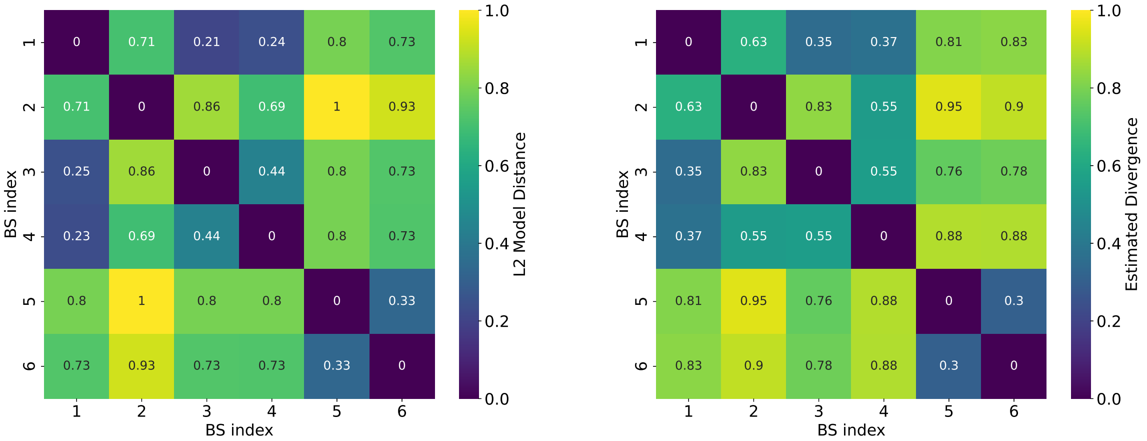
To achieve better performance without increasing pilot overhead or storage cost, data augmentation is conducted for each local dataset. As shown in Fig. 5.8(a), when the augmentation time $A = 4$ (i.e., expanding the dataset from 200 to 800 batches at each BS), the average test BER of all neural receivers achieves 1 dB gain at BER $= 10^{-5}$ compared with that in the unaugmented case. However, when the augmentation time increases to 8, the BER performance drops as intensive augmentation may cause the overfitting problem. Similar patterns can also be observed in Fig. 5.8(b) where inter-cell interference is considered, and a significant gain of around 3 dB is achieved at BER $= 10^{-3}$ compared with the unaugmented case. Therefore, in GraphRx, data augmentation with $A = 4$ is adopted to enrich each local dataset, which reduces the pilot overhead by 75% compared with collecting four times the current quantity of data.

5.5.3 Effectiveness of Divergence Approximation

Due to the unavailability of theoretical bound in practice, ℓ_2 model distance is used to approximate pairwise divergence d_{mj} that incorporates $\mathcal{H}\Delta\mathcal{H}$-divergence and λ_{mj}, i.e., $d_{mj} = d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_m, \mathcal{D}_j) + 2\lambda_{mj}$. In domain adaptation tasks, λ_{mj} is generally small and thus can be ignored. Following the common method in Ben-David et al. (2010), $\mathcal{H}\Delta\mathcal{H}$-divergence is empirically estimated by training a non-linear classifier to discriminate between any pair of local datasets. The classification accuracy between [0,1] is deemed as the estimated divergence between two datasets. Therefore, $\binom{M}{2}$ classifiers in total are trained to obtain the estimated divergence between



(a) ℓ_2 model distance (Hetero-1) (b) $\mathcal{H}\Delta\mathcal{H}$-divergence (Hetero-1)

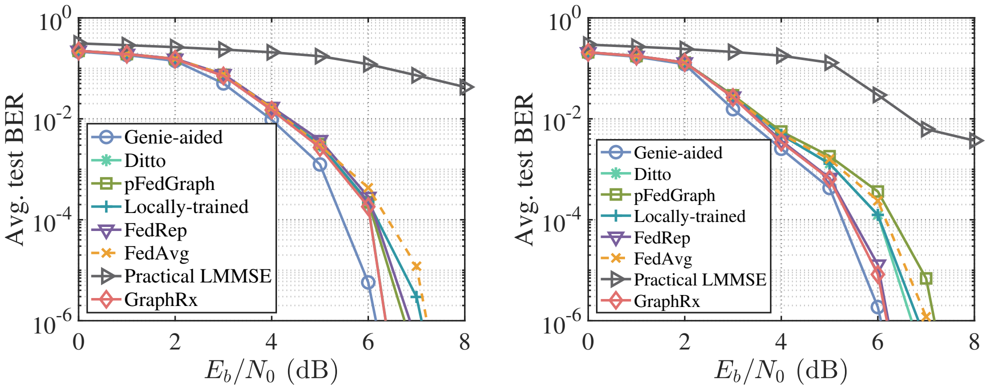


(c) ℓ_2 model distance (Hetero-2) (d) $\mathcal{H}\Delta\mathcal{H}$-divergence (Hetero-2)

Figure 5.9: Correlations between model distance and estimated divergence.

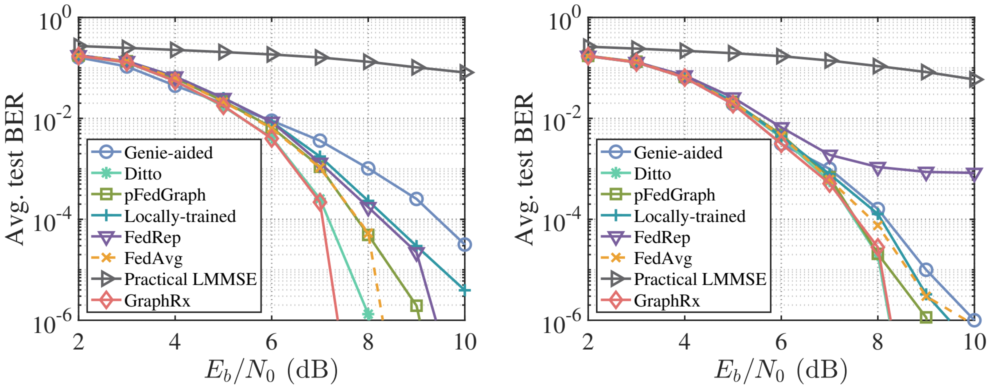
each pair among M BSs. As shown in Fig. 5.9, the distances between converged models are compared with the empirical $\mathcal{H}\Delta\mathcal{H}$-divergence between datasets of all BSs. There exists a strong positive correlation between each BS pair's model distance and distribution divergence under both Hetero-1 and Hetero-2. Taking BS 1 shown in Fig. 5.9(a) and 5.9(b) as an example, BS pair (1,2) has both the smallest distance and divergence at the same time, and BS 1 exhibits large distances and divergences simultaneously from BSs 3 and 4. Such positive correlations are observed by comparing each row of the distance matrix and divergence matrix in Fig. 5.9.

20011372



(a) Hetero-1 (b) Hetero-2

Figure 5.10: Coded BER tested without inter-cell interference.



(a) Hetero-1 (b) Hetero-2

Figure 5.11: Coded BER tested with inter-cell interference.

5.5.4 Comparison with Baselines

The test performance of GraphRx is evaluated and compared with other baseline receivers. For neural receivers, data augmentation with $A = 4$ is adopted for all the learning algorithms for fair comparison. The SNR gain is measured in dB at BER $= 10^{-5}$ unless stated otherwise.

First, we consider the scenario without inter-cell interference as shown in Fig. 5.10 for Hetero-1 and Hetero-2. As observed in both figures, the genie-aided LMMSE receiver serves as the performance upper bound in the interference-free case, outperforming GraphRx by 0.4 dB for Hetero-1 and 0.2 dB for Hetero-2. Our GraphRx achieves the best BER performance among all the other baselines under these two heterogeneous distributions. Specifically, for Hetero-1, GraphRx exceeds both pFedGraph and FedRep by around 0.4 dB and it outperforms FedAvg by 0.9 dB. Note that the performance of Ditto is very close to GraphRx for Hetero-1, while Ditto

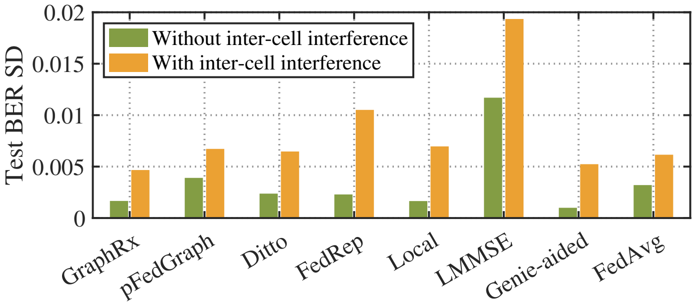


Figure 5.12: Standard deviation of coded BER tested for Hetero-2.

performs 0.4 dB worse than GraphRx for Hetero-2 shown in Fig. 5.10(b). In addition, under both Hetero-1 and Hetero-2, FedAvg is inferior to local learning by around 0.2 dB, which reflects that learning a unified global model is not optimal for improving local performance when data are heterogeneously distributed. It is also worth noting that all the methods based on neural receivers can outperform the practical LMMSE receiver by a significant margin of over 4 dB.

Next, we consider the scenario with inter-cell interference shown in Fig. 5.11. Under both Hetero-1 and Hetero-2, the genie-aided LMMSE receiver is no longer the performance upper bound since it does not have any knowledge of the interfering channels. Under Hetero-1 illustrated in Fig. 5.11(a), GraphRx achieves notable performance gains over all the listed baselines. Specifically, it outperforms three other PFL algorithms (i.e., Ditto, pFedGraph, and FedRep) by 0.5 dB, 1.5 dB, and 2.1 dB, respectively. Interestingly, the performance of FedAvg is 1.6 dB better than that of local learning, which suggests that even a non-personalized aggregate model can achieve a substantial gain when inter-cell interference exists. Under Hetero-2 shown in Fig. 5.11(b), GraphRx performs equally well as Ditto, but outperforms all other baselines. Two additional insights are drawn: 1) GraphRx is robust to interference, while the performance of FedRep varies significantly; 2) when interference exists, collaboration learning seems more challenging under Hetero-2 than Hetero-1, as Hetero-2 has a higher heterogeneity level in both local distributions and data quantities than Hetero-1. This issue is subject to future research.

Finally, the fairness among six BSs is presented in Fig. 5.12, which is quantified by the standard deviation (SD) of test BERs of all the neural receivers. The SD is evaluated at SNR $=$ 5dB for Hetero-2 that has a higher heterogeneity level. As shown in Fig. 5.12, GraphRx achieves the lowest BER SD whether interference exists or not, which means the best fairness can be guaranteed among all collaborators. In addition, with inter-cell interference, the variation in the performance of neural receivers becomes larger.

5.6 Summary

A multi-cell collaborative learning scheme named GraphRx was developed in this chapter. With a weighted graph capturing collaboration relationships among BSs, GraphRx was formulated as a personalized federated learning problem. To solve this problem, the personalized models were updated locally at each BS and the collaboration graph was optimized centrally at the server by minimizing an approximate generalization bound, and these two steps were performed alternatively. Data augmentation techniques were further incorporated into GraphRx to reduce overhead of training pilot signals. GraphRx achieved significant performance gains over the baselines.

20011372

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The dissertation focused on developing AI-assisted algorithms to boost network throughput in future 6G wireless networks, by addressing critical problems at three architectural levels: the network infrastructure level, resource management level, and link level. The key contributions of this dissertation are elaborated in the following.

At the network infrastructure level, the problem of augmenting multi-hop backhaul architecture with relays to boost network throughput was studied. A clique-based bottleneck theory was developed for general wireless networks. In this theory, the clique-based bottleneck structures were constructed to identify the complicated interactions among bottlenecks and determine the relationship between the network architecture and the network throughput. A clique-gradient computation algorithm was designed based on the constructed bottleneck structure to quantify how the perturbation of each clique impacted the overall network throughput. A DRL-based relay placement approach named *DeepRP* was designed by incorporating the clique-based bottleneck theory, to augment the 6G backhaul architecture for higher network throughput. This approach significantly boosted network throughput as compared with the baselines.

At the resource management level, the problem of RAN slice enforcement was studied. Fine-grained slice enforcement was conducted at the link level. It was formulated as an optimization problem that satisfied three key requirements of RAN slicing, i.e., soft isolation under both balanced and unbalanced slicing policies, users' QoS guarantee, and long-term conformance of slicing policies. The interference management with link-level granularity was conducted in multi-cell RAN slicing. To tackle the coupling between the long-term objective and short-term scheduling, a DRL-based heuristic approach was developed for the slice enforcement problem.

Without depending on the prior knowledge of the time-varying channels and interference conditions, the DRL-based heuristic approach adapted to the dynamic environment and achieved a near-optimal solution via online learning. Following the DRL-based heuristic approach, *LinkSlice* was designed as an iterative slice enforcement algorithm consisting of two stages. The first stage determined the transmission rates by capturing link interference via an interference-graph-based GNN. Given the transmission rates, the second stage allocated PRBs for each slice by considering QoS requirements and soft slice isolation. LinkSlice highly outperformed existing schemes.

At the link level, two critical problems regarding online learning of neural receivers were thoroughly investigated. The developed approaches at this level provided high link-level throughput, which directly leads to improvement of network throughput.

First, the problem of efficient short-term online adaptation for neural channel estimators was addressed. An SSL task was introduced on top of the original channel-estimation task to facilitate label-free adaptation of neural channel estimators. It was designed as reconstructing the randomly-masked portions of received radio frames. An online symbol-recovery mechanism was designed to provide estimated data-symbols as the task input component, without incurring extra pilot overhead. A two-branch MAE model architecture named *ChannelMAE* was designed to consolidate the SSL task with the channel-estimation task. With each branch dedicated to one task, the two branches shared a feature encoder but had their task-specific decoders. An offline-online two-phase training strategy was designed for ChannelMAE. Particularly in the online adaptation phase, the shared encoder was adapted by optimizing the SSL branch with batch-wise online learning. The adapted encoder improved online channel-estimation performance by implicitly capturing channel statistical features. Extensive experiments were carried out to validate key parameters and mechanisms of ChannelMAE. Performance results showed that ChannelMAE significantly outperformed the state-of-the-art adaptation schemes.

Second, the problem of long-term multi-cell collaborative learning for end-to-end neural receivers was studied. A collaborative learning scheme, called *GraphRx*, was proposed based on a collaboration graph among BSs to retrain uplink neural receivers in multiple cells together. It acquired channel knowledge in multiple cells collaboratively and handled multi-cell interference effectively. GraphRx was formulated as a PFL problem, wherein the personalized models of neural receivers and the collaboration graph were optimized jointly to achieve tradeoff between generalization and personalization. The solution to the PFL problem was derived and was mapped to the federated learning paradigm as an iterative algorithm alternating between the server and BSs. Particularly, an approximate generalization bound was derived so that the server could determine the collaboration graph without relying on local data of BSs. Performance results showed that GraphRx significantly

20011372

outperformed the state-of-the-art schemes.

In conclusion, this dissertation pushed forward the research frontier, by providing a comprehensive view of developing AI algorithms in wireless networking at different architectural levels. It investigated four critical but under-explored research problems, where diverse machine learning paradigms such as SL, SSL, and DRL were adopted.

The innovation and impact of this dissertation were further concluded in the aspects of theory and algorithms. From the theory aspect, this dissertation presented two key theoretical contributions. First, a novel clique-based bottleneck theory framework was first developed to systematically identify network bottlenecks and derive fairness-optimized network throughput. This framework served as both a versatile analytical tool and a fundamental design guideline for next-generation wireless network planning. Second, a theoretical framework was developed to leverage generalization bounds for designing personalized federated learning systems. The theoretical frameworks developed in this dissertation contributed to both wireless networking and machine learning. From the algorithm aspect, this dissertation developed a variety of AI-assisted algorithms. Specifically, the following innovative algorithms were designed: 1) a DRL-based relay placement algorithm (DeepRP) to optimize backhaul network infrastructure; 2) a DRL-based network slice enforcement algorithm (LinkSlice) for management resources in a multi-cell RAN; 3) an SSL-based online adaptation algorithm (ChannelMAE) for facilitating efficient online adaptation of neural channel estimators; 4) a graph-based PFL algorithm (GraphRx) for improving the generalization ability of neural receivers in a multi-cell network. These AI-assisted algorithms were developed to boost network throughput for next-generation wireless networks. Meanwhile, there also exists great potential that they can be adapted to other problems, such as network planning, resource scheduling, and many other network optimization tasks.

6.2 Future Work

This dissertation presented comprehensive studies across three architectural levels, yet several avenues remain open for deeper exploration.

For Chapter 2, two aspects of DeepRP require further investigation. First, the performance guarantee of DeepRP relied on the clique-based bottleneck theory. More theoretical analysis on the connection between the developed theory and DeepRP will be pursued in future work. Second, for general wireless mesh networks beyond backhaul settings, deterministic flow states may be difficult to capture. Relay placement should therefore target statistical flow states with random source–destination pairs, and how to achieve higher throughput in such scenarios remains an important open problem.

For Chapter 3, three interesting topics need to be further investigated. First, how to construct interference graphs was not studied in this work. Although related literature exists, obtaining an interference graph accurately and promptly remains challenging. Second, spatial reuse was not considered. In beyond-5G networks, beamforming and multi-user MIMO are common physical-layer techniques, and integrating LinkSlice with these technologies requires further research. Third, joint optimization of user admission, power allocation, and time/frequency allocation was not addressed. While LinkSlice incorporated a simple admission control scheme, it did not guarantee fairness among users. Designing a unified framework for fair admission and power allocation within LinkSlice is thus a meaningful future direction.

For Chapter 4, the designed MAE model architecture of ChannelMAE requires further studies to support multi-antenna communications. Model pruning techniques can be applied to reduce model size, and continual learning can be incorporated to mitigate catastrophic forgetting issues. Furthermore, extending the SSL-assisted two-task framework to a broader range of wireless applications represents another important research direction.

For Chapter 5, GraphRx was built on an existing neural receiver architecture, which itself requires further improvement. First, an architecture that eliminates the need for DMRS signals is highly desirable. Second, how to extend neural receivers to support MIMO communications remains an open question. In addition, handling scenarios with both inter-cell interference and high cell heterogeneity requires further exploration.

Taken together, the four research works highlight two overarching directions for future investigation. First, optimizing the three architectural levels in isolation may lead to suboptimal performance. Coordinating and jointly optimizing designs across different levels to exploit cross-layer synergies remains a key challenge. Second, throughput alone cannot fully characterize the performance of next-generation networks. Extending the proposed approaches to incorporate other critical metrics, such as end-to-end latency and reliability, will be essential to meet the stringent requirements of future wireless networks.

20011372

Bibliography

3GPP TR 38.801 (2017). Study on new radio access technology: Radio access architecture and interfaces. Technical report, 3rd Generation Partnership Project. V14.0.0.

3GPP TR 38.802 (2017). Study on new radio access technology Physical layer aspects. Technical report, 3rd Generation Partnership Project. V14.2.0.

3GPP TS 28.541 (2024). Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3. Technical report, 3rd Generation Partnership Project. V18.3.0.

3GPP TS 28.542 (2018). Management and orchestration of networks and network slicing; 5G Core Network (5GC) Network Resource Model (NRM). Technical report, 3rd Generation Partnership Project. V1.1.0.

3GPP TS 38.133 (2024). Requirements for support of radio resource management. Technical report, 3rd Generation Partnership Project. V18.8.0.

3GPP TS 38.211 (2024). Physical channels and modulation. Technical report, 3rd Generation Partnership Project. V18.5.0.

3GPP TS 38.214 (2021). Physical layer procedures for data. Technical report, 3rd Generation Partnership Project. V18.3.0.

3GPP TS 38.901 (2024). Study on channel model for frequencies from 0.5 to 100 GHz. Technical report, 3rd Generation Partnership Project. V18.0.0.

6G Flagship (2025). 6G flagship's vision for 2030. `https://www.6gflagship.com/`.

Abdelmoaty, A., D. Naboulsi, G. Dahman, G. Poitau, and F. Gagnon (2022). Resilient topology design for wireless backhaul: A deep reinforcement learning approach. *IEEE Wireless Communications Letters 11*(12), 2532–2536.

Addad, R. A., D. L. C. Dutra, T. Taleb, and H. Flinck (2021). Toward using reinforcement learning for trigger selection in network slice mobility. *IEEE Journal on Selected Areas in Communications 39*(7), 2241–2253.

Afolabi, I., T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck (2018). Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys and Tutorials 20*(3), 2429–2453.

Aoudia, F. A. and J. Hoydis (2021). End-to-end learning for OFDM: From neural receivers to pilotless communication. *IEEE Transactions on Wireless Communiations 21*(2), 1049–1063.

Aoudia, F. A. and J. Hoydis (2022). Waveform learning for next-generation wireless communication systems. *IEEE Transactions on Communications 70*(6), 3804–3817.

Azimi, Y., S. Yousefi, H. Kalbkhani, and T. Kunz (2022). Energy-efficient deep reinforcement learning assisted resource allocation for 5g-ran slicing. *IEEE Transactions on Vehicular Technology 71*(1), 856–871.

Belgiovine, M., K. Sankhe, C. Bocanegra, D. Roy, and K. R. Chowdhury (2021). Deep learning at the edge for channel estimation in beyond-5G massive mimo. *IEEE Wireless Communications 28*(2), 19–25.

Bello, I., H. Pham, Q. V. Le, M. Norouzi, and S. Bengio (2017). Neural combinatorial optimization with reinforcement learning. In *Proceeding of International Conference on Learning Representations Workshop*, pp. 1–5.

Belotti, P., C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan (2013). Mixed-integer nonlinear optimization. *Acta Numerica 22*, 1–131.

Ben-David, S., J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan (2010). A theory of learning from different domains. *Machine learning 79*, 151–175.

Boyd, S. and L. Vandenberghe (2004). *Convex Optimization.* Cambridge University Press.

Chang, C.-Y., C.-T. Chang, T.-C. Wang, and M.-H. Li (2014). Throughput-enhanced relay placement mechanism in wimax 802.16 j multihop relay networks. *IEEE Systems Journal 9*(3), 728–742.

Chang, R. Y., Z. Tao, J. Zhang, and C.-C. J. Kuo (2009). Multicell OFDMa downlink resource allocation using a graphic framework. *IEEE Transactions on Vehicular Technology 58*(7), 3494–3507.

Chao, I.-F. and W.-S. Hsu (2023). A MAC protocol design for maximizing end-to-end throughput and fairness guarantee in chain-based multi-hop wireless backhaul networks. *IEEE Transactions on Mobile Computing 22*(7), 3743–3756.

Chen, J., U. Mitra, and D. Gesbert (2021). 3D urban UAV relay placement: Linear complexity algorithm and analysis. *IEEE Transactions on Wireless Communications 20*(8), 5243–5257.

Chen, X., Z. Zhao, C. Wu, M. Bennis, H. Liu, Y. Ji, and H. Zhang (2019). Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach. *IEEE Journal on Selected Areas in Communications 37*(10), 2377–2392.

Chen, Z., F. Gu, and R. Jiang (2020). Channel estimation method based on transformer in high dynamic environment. In *Proceedings of 2020 International Conference on Wireless Communications and Signal Processing*, pp. 817–822.

Cheng, J., Y. Ke, A. W.-C. Fu, J. X. Yu, and L. Zhu (2011). Finding maximal cliques in massive networks. *ACM Transactions on Database Systems 36*(4), 1–34.

Cho, Y. S., J. Kim, W. Y. Yang, and C. G. Kang (2010). *MIMO-OFDM Wireless Communications with MATLAB.* John Wiley and Sons.

Collins, L., H. Hassani, A. Mokhtari, and S. Shakkottai (2021). Exploiting shared representations for personalized federated learning. In *Proceedings of International Conference on Machine Learning*, pp. 2089–2099.

Diamanti, M., P. Charatsaris, E. E. Tsiropoulou, and S. Papavassiliou (2021). The prospect of reconfigurable intelligent surfaces in integrated access and backhaul networks. *IEEE Transactions on Green Communications and Networking 6*(2), 859–872.

Dong, T., Z. Zhuang, Q. Qi, J. Wang, H. Sun, F. R. Yu, T. Sun, C. Zhou, and J. Liao (2021). Intelligent joint network slicing and routing via gcn-powered multi-task deep reinforcement learning. *IEEE Transactions on Cognitive Communications and Networking 8*(2), 1269–1286.

D'Oro, S., F. Restuccia, and T. Melodia (2020). Toward operator-to-waveform 5G radio access network slicing. *IEEE Communications Magazine 58*(4), 18–23.

D'Oro, S., F. Restuccia, T. Melodia, and S. Palazzo (2018). Low-complexity distributed radio access network slicing: Algorithms and experimental results. *IEEE/ACM Transactions on Networking 26*(6), 2815–2828.

D'Oro, S., F. Restuccia, A. Talamonti, and T. Melodia (2019). The slice is served: Enforcing radio access network slicing in virtualized 5G systems. In *Proceedings of IEEE Conference on Computer Communications*, pp. 442–450.

Eisen, M. and A. Ribeiro (2020). Optimal wireless resource allocation with random edge graph neural networks. *IEEE Transactions on Signal Processing 68*, 2977–2991.

Eslami Rasekh, M., D. Guo, and U. Madhow (2020). Joint routing and resource allocation for millimeter wave picocellular backhaul. *IEEE Transactions on Wireless Communications 19*(2), 783–794.

Farsad, N. and A. Goldsmith (2018). Neural network detection of data sequences in communication systems. *IEEE Transactions on Signal Processing 66*(21), 5663–5678.

Feriani, A., D. Wu, S. Liu, and G. Dudek (2023). Cebed: A benchmark for deep data-driven OFDM channel estimation. *arXiv preprint arXiv:2306.13761*.

Ferrus, R., O. Sallent, J. Perez-Romero, and R. Agusti (2018). On 5G radio access network slicing: Radio interface protocol features and configuration. *IEEE Communications Magazine 56*(5), 184–192.

Fischer, M. B., S. Dörner, S. Cammerer, T. Shimizu, H. Lu, and S. Ten Brink (2022). Adaptive neural network-based OFDM receivers. In *Proceedings of IEEE International Workshop on Signal Processing Advances in Wireless Communications*, pp. 1–5.

Fischer, M. B., S. Dörner, F. Krieg, S. Cammerer, and S. ten Brink (2022). Adaptive NN-based OFDM receivers: Computational complexity vs. achievable performance. In *Proceedings of IEEE Asilomar Conference on Signals, Systems, and Computers*, pp. 194–199.

Flushing, E. F. and G. A. D. Caro (2013). A flow-based optimization model for throughput-oriented relay node placement in wireless sensor networks. In *Proceedings of ACM Symposium on Applied Computing*, pp. 632–639.

20011372

Foukas, X., G. Patounas, A. Elmokashfi, and M. K. Marina (2017). Network slicing in 5G: Survey and challenges. *IEEE Communications Magazine 55*(5), 94–100.

Gandelsman, Y., Y. Sun, X. Chen, and A. Efros (2022). Test-time training with masked autoencoders. *Proceedings of Advances in Neural Information Processing Systems 35*, 29374–29385.

Gao, H., Z. Wang, and S. Ji (2018). Large-scale learnable graph convolutional networks. In *Proceedings of ACM International Conference on Knowledge Discovery*, pp. 1416–1424.

Gao, J. and L. Zhang (2006). Load-balanced short-path routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems 17*(4), 377–388.

Giordani, M., M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi (2020, March). Toward 6G networks: Use cases and technologies. *IEEE Communications Magazine 58*(3), 55–61.

Guo, T. and A. Suárez (2019). Enabling 5G RAN slicing with EDF slice scheduling. *IEEE Transactions on Vehicular Technology 68*(3), 2865–2877.

Gurobi Optimization, LLC (2021). Gurobi Optimizer Reference Manual.

He, K., X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick (2022). Masked autoencoders are scalable vision learners. In *Proceedings of IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009.

He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *Proceedings of IEEE/CVF conference on computer vision and pattern recognition(CVPR)*, pp. 770–778.

Henderson, P., R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger (2018). Deep reinforcement learning that matters. In *Proceedings of AAAI Conference on Artificial Intelligence*, pp. 3207–3214.

Honkala, M., D. Korpi, and J. M. Huttunen (2021). DeepRx: Fully convolutional deep learning receiver. *IEEE Transactions on Wireless Communiations 20*(6), 3925–3940.

Hoydis, J. et al. (2023). Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling. *arXiv preprint arXiv:2303.11103*.

Hoydis, J., S. Cammerer, F. A. Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller (2022). Sionna: An open-source library for next-generation physical layer research. *arXiv preprint arXiv:2203.11854*.

Hu, Q. and D. M. Blough (2017). Relay selection and scheduling for millimeter wave backhaul in urban environments. In *Proceedings of IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pp. 206–214.

Hua, Y., R. Li, Z. Zhao, X. Chen, and H. Zhang (2020). GAN-powered deep distributional reinforcement learning for resource management in network slicing. *IEEE Journal on Selected Areas in Communications 38*(2), 334–349.

Huang, C. and X. Wang (2023). A Bayesian approach to the design of backhauling topology for 5G IAB networks. *IEEE Transactions on Mobile Computing 22*(4), 1867–1879.

20011372

Huang, N., P.-C. Hsieh, K.-H. Ho, and I.-C. Wu (2024). PPO-Clip attains global optimality: Towards deeper understandings of clipping. In *Proceedings of AAAI Conference on Artificial Intelligence*, Volume 38, pp. 12600–12607.

Huang, S. and S. Ontañón (2020). A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171*.

Huang, X. L. and B. Bensaou (2001). On max-min fairness and scheduling in wireless ad-hoc networks: Analytical framework and implementation. In *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 221–231.

Jain, K., J. Padhye, V. N. Padmanabhan, and L. Qiu (2003). Impact of interference on multi-hop wireless network performance. In *Proceedings of International Conference on Mobile Computing and Networking*, pp. 66–80.

Jiang, J., C. Dun, T. Huang, and Z. Lu (2020). Graph convolutional reinforcement learning. In *Proceedings of 8th International Conference on Learning Representations*, pp. 1–12.

Jiang, P., T. Wang, B. Han, X. Gao, J. Zhang, C.-K. Wen, S. Jin, and G. Y. Li (2021). AI-aided online adaptive OFDM receiver: Design and experimental results. *IEEE Transactions on Wireless Communiations 20*(11), 7655–7668.

Kipf, T. N. and M. Welling (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kong, L., X. Liu, X. Zhang, J. Xiong, H. Zhao, and J. Wei (2025). Representation-based continual learning for channel estimation in dynamic wireless environments. *IEEE Transactions on Wireless Communications*.

Ksentini, A. and N. Nikaein (2017). Toward enforcing network slicing on RAN: Flexibility and resources abstraction. *IEEE Communications Magazine 55*(6), 102–108.

Lee, M., G. Yu, and G. Y. Li (2021). Graph embedding-based wireless link scheduling with few training samples. *IEEE Transactions on Wireless Communications 20*(4), 2282–2294.

Letaief, K. B., W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang (2019, August). The roadmap to 6G: AI empowered wireless networks. *IEEE Communications Magazine 57*(8), 84–90.

Li, J., Y. Niu, H. Wu, B. Ai, R. He, N. Wang, and S. Chen (2023). Joint optimization of relay selection and transmission scheduling for UAV-aided mmwave vehicular networks. *IEEE Transactions on Vehicular Technology 72*(5), 6322–6334.

Li, L., H. Chen, H.-H. Chang, and L. Liu (2020). Deep residual learning meets OFDM channel estimation. *IEEE Wireless Communications Letters 9*(5), 615–618.

Li, R. and P. Patras (2020). Max-min fair resource allocation in millimetre-wave backhauls. *IEEE Transactions on Mobile Computing 19*(8), 1879–1895.

Li, R., C. Wang, Z. Zhao, R. Guo, and H. Zhang (2020). The lstm-based advantage actor-critic learning for resource management in network slicing with user mobility. *IEEE Communications Letters 24*(9), 2005–2009.

Li, T., S. Hu, A. Beirami, and V. Smith (2021). Ditto: Fair and robust federated learning through personalization. In *Proceedings of International Conference on Machine Learning*, pp. 6357–6368.

Liang, L., H. Ye, G. Yu, and G. Y. Li (2020). Deep-learning-based wireless resource allocation with application to vehicular networks. *Proceedings of IEEE 108*(2), 341–356.

Liang, W., C. Ma, M. Zheng, and L. Luo (2021). Relay node placement in wireless sensor networks: From theory to practice. *IEEE Transactions on Mobile Computing 20*(4), 1602–1613.

Lim, B., S. Son, H. Kim, S. Nah, and K. Mu Lee (2017). Enhanced deep residual networks for single image super-resolution. In *Proceedings of IEEE conference on computer vision and pattern recognition workshops*, pp. 136–144.

Lin, C. and G. Y. Li (2015). Indoor terahertz communications: How many antenna arrays are needed? *IEEE Transactions on Wireless Communiations 14*(6), 3097–3107.

Liu, B., Q. Cai, Z. Yang, and Z. Wang (2019). Neural trust region/proximal policy optimization attains globally optimal policy. In *Proceedings of Advances in Neural Information Processing Systems*, Volume 32.

Liu, W., J. He, and S. Chang (2010). Large graph construction for scalable semi-supervised learning. In *Proceedings of International Conference on Machine Learning*, pp. 679–686.

Liu, Y., B. Dai, Z. Han, and H. V. Poor (2023). Deep reinforcement learning for resource allocation in 6g wireless networks. *IEEE Network 37*(2), 54–61.

Liu, Y., P. Kothari, B. van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi (2021). TTT++: When does self-supervised test-time training fail or thrive? In *Proceedings of Advances in Neural Information Processing Systems*, Volume 34, pp. 21808–21820.

Liu, Y., Z. Tan, H. Hu, L. J. Cimini, and G. Y. Li (2014). Channel estimation for OFDM. *IEEE Communications Surveys and Tutorials 16*(4), 1891–1908.

Liu, Y.-J., G. Feng, Y. Sun, S. Qin, and Y.-C. Liang (2020). Device association for ran slicing based on hybrid federated deep reinforcement learning. *IEEE Transactions on Vehicular Technology 69*(12), 15731–15745.

Luan, D. and J. Thompson (2022). Attention based neural networks for wireless channel estimation. In *Proceedings of IEEE Vehicular Technology Conference (Spring)*, pp. 1–5. IEEE.

Luan, D. and J. S. Thompson (2023). Channelformer: Attention based neural solution for wireless channel estimation and effective online training. *IEEE Transactions on Wireless Communications 22*(10), 6562–6577.

López-Pérez, D., X. Chu, A. V. Vasilakos, and H. Claussen (2013). On distributed and coordinated resource allocation for interference mitigation in self-organizing lte networks. *IEEE/ACM Transactions on Networking 21*(4), 1145–1158.

Magán-Carrión, R., R. A. Rodríguez-Gómez, J. Camacho, and P. García-Teodoro (2016). Optimal relay placement in multi-hop wireless networks. *Ad Hoc Networks 46*, 23–36.

Mandelli, S., M. Andrews, S. Borst, and S. Klein (2019). Satisfying network slicing constraints via 5g mac scheduling. In *Proceedings of IEEE Conference on Computer Communications*, pp. 2332–2340.

Mashhadi, M. B., N. Shlezinger, Y. C. Eldar, and D. Gündüz (2021). Fedrec: Federated learning of universal receivers over fading channels. In *Proceedings of IEEE Statistical Signal Processing Workshop*, pp. 576–580. IEEE.

McMahan, B., E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas (2017). Communication-efficient learning of deep networks from decentralized data. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282.

Mei, J., X. Wang, K. Zheng, G. Boudreau, A. B. Sediq, and H. Abou-Zeid (2021). Intelligent radio access network slicing for service provisioning in 6G: A hierarchical deep reinforcement learning approach. *IEEE Transactions on Communications 69*(9), 6063–6078.

Minelli, M., M. Ma, M. Coupechoux, J.-M. Kelif, M. Sigelle, and P. Godlewski (2014). Optimal relay placement in cellular networks. *IEEE Transactions on Wireless Communiations 13*(2), 998–1009.

Niknam, S., H. S. Dhillon, and J. H. Reed (2020). Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine 58*(6), 46–51.

Nikolov, M. and Z. J. Haas (2016). Relay placement in wireless networks: Minimizing communication cost. *IEEE Transactions on Wireless Communiations 15*(5), 3587–3602.

Ntontin, K., D. Selimis, A.-A. A. Boulogeorgos, A. Alexandridis, A. Tsolis, V. Vlachodimitropoulos, and F. Lazarakis (2021). Optimal reconfigurable intelligent surface placement in millimeter-wave communications. In *Proceedings of the European Conference on Antennas and Propagation*, pp. 1–5.

NVIDIA (2025). SionnaRT: A lightning-fast stand-alone ray tracer for radio propagation modeling. `https://nvlabs.github.io/sionna/`. Accessed September 2, 2025.

OpenStreetMap contributors (2025). Building footprints. `https://openstreetmap.org`. Accessed September 2, 2025.

Park, S., H. Jang, O. Simeone, and J. Kang (2021). Learning to demodulate from few pilots via offline and online meta-learning. *IEEE Transactions on Signal Processing 69*, 226–239.

Paternain, S., L. F. O. Chamon, M. Calvo-Fullana, and A. Ribeiro (2019). Constrained reinforcement learning has zero duality gap. In *Proceedings of Advances in Neural Information Processing Systems*, pp. 7553–7563.

Pihlajasalo, J., D. Korpi, M. Honkala, J. M. Huttunen, T. Riihonen, J. Talvitie, A. Brihuega, M. A. Uusitalo, and M. Valkama (2023). Deep learning OFDM receivers for improved power efficiency and coverage. *IEEE Transactions on Wireless Communiations 22*(8), 5518–5535.

Polese, M., L. Bonati, S. D'oro, S. Basagni, and T. Melodia (2023). Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges. *IEEE Communications Surveys and Tutorials 25*(2), 1376–1411.

Rahman, M. and H. Yanikomeroglu (2010). Enhancing cell-edge performance: a downlink dynamic interference avoidance scheme with inter-cell coordination. *IEEE Transactions on Wireless Communications 9*(4), 1414–1425.

Raviv, T., S. Park, O. Simeone, Y. C. Eldar, and N. Shlezinger (2023). Online meta-learning for hybrid model-based deep receivers. *IEEE Transactions on Wireless Communications 22*(10), 6415–6431.

Ros-Giralt, J., N. Amsel, S. Yellamraju, J. Ezick, R. Lethin, Y. Jiang, A. Feng, and L. Tassiulas (2022, October). A quantitative theory of bottleneck structures for data networks. *arXiv preprint arVix:2210.03534*.

Ros-Giralt, J., N. Amsel, S. Yellamraju, J. Ezick, R. Lethin, Y. Jiang, A. Feng, L. Tassiulas, Z. Wu, M. Y. Teh, and K. Bergman (2021). Designing data center networks using bottleneck structures. In *Proceedings of ACM Special Interest Group on Data Communication*, pp. 319–348.

Ros-Giralt, J., A. Bohara, S. Yellamraju, M. H. Langston, R. Lethin, Y. Jiang, L. Tassiulas, J. Li, Y. Tan, and M. Veeraraghavan (2019, December). On the bottleneck structure of congestion-controlled networks. In *Proceedings of ACM Special Interest Group on Measurement and Evaluation*, pp. 1–31.

Saha, C. and H. S. Dhillon (2019). Millimeter wave integrated access and backhaul in 5G: Performance analysis and design insights. *IEEE Journal on Selected Areas in Communications 37*(12), 2669–2684.

Saqib, N. U., S. Hou, S. H. Chae, and S.-W. Jeon (2024). Reconfigurable intelligent surface aided hybrid beamforming: Optimal placement and beamforming design. *IEEE Transactions on Wireless Communiations*.

Schulman, J., P. Moritz, S. Levine, M. Jordan, and P. Abbeel (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sexton, C., N. Marchetti, and L. A. DaSilva (2020). On provisioning slices and overbooking resources in service tailored networks of the future. *IEEE/ACM Transactions on Networking 28*(5), 2106–2119.

Shi, D., L. Li, T. Ohtsuki, M. Pan, Z. Han, and H. V. Poor (2021). Make smart decisions faster: Deciding d2d resource allocation via stackelberg game guided multi-agent deep reinforcement learning. *IEEE Transactions on Mobile Computing 21*(12), 4426–4438.

Simsek, M., O. Orhan, M. Nassar, O. Elibol, and H. Nikopour (2021). IAB topology design: A graph embedding and deep reinforcement learning approach. *IEEE Communications Letters 25*(2), 489–493.

Soltani, M., V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh (2019). Deep learning-based channel estimation. *IEEE Communications Letters 23*(4), 652–655.

Soltani, M., V. Pourahmadi, and H. Sheikhzadeh (2020). Pilot pattern design for deep learning-based channel estimation in OFDM systems. *IEEE Wireless Communications Letters 9*(12), 2173–2176.

Sun, Y., S. Qin, G. Feng, L. Zhang, and M. A. Imran (2021). Service provisioning framework for RAN slicing: User admissibility, slice association and bandwidth allocation. *IEEE Transactions on Mobile Computing 20*(12), 3409–3422.

Sun, Y., X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt (2020). Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of International Conference on Machine Learning*, pp. 9229–9248.

Tang, J., B. Shim, and T. Q. Quek (2019). Service multiplexing and revenue maximization in sliced c-ran incorporated with urllc and multicast embb. *IEEE Journal on Selected Areas in Communications 37*(4), 881–895.

Toh, C. K., A.-n. Le, and Y.-z. Cho (2009). Load balanced routing protocols for ad hoc mobile wireless networks. *IEEE Communications Magazine 47*(8), 78–84.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017). Attention is all you need. In *Proceedings of International Conference on Neural Information Processing Systems*, pp. 6000–6010.

Wang, O., J. Gao, and G. Y. Li (2022). Learn to adapt to new environments from past experience and few pilot blocks. *IEEE Transactions on Cognitive Communications and Networking 9*(2), 373–385.

Wang, P., H. Jiang, W. Zhuang, and H. V. Poor (2008). Redefinition of max-min fairness in multi-hop wireless networks. *IEEE Transactions on Wireless Communications 7*(12), 4786–4791.

Wang, T., S. Chen, Y. Zhu, A. Tang, and X. Wang (2022). Linkslice: Fine-grained network slice enforcement based on deep reinforcement learning. *IEEE Journal on Selected Areas in Communications 40*(8), 2378–2394.

Wang, T., S. Wang, Y. G. Li, and X. Wang (2024). Collaborative learning for less online retraining of neural receivers. In *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing.*

Wang, T., S. Wang, Y. G. Li, and X. Wang (2025a). Collaborative learning for less online retraining of neural receivers. In *Proceedings of IEEE International Conference on Computer Communications.*

Wang, T., S. Wang, Y. G. Li, and X. Wang (2025b). Collaborative learning for less online retraining of neural receivers. In *Proceedings of IEEE International Conference on Computer Communications.*

Wang, T. and X. Wang (2023). Boosting capacity for 6g terahertz mesh networks based on bottleneck structures. In *Proceedings of IEEE Global Communications Conference*, pp. 4589–4594.

Wang, Y., X. Bai, Y. Yuan, M. Chen, Q. Yang, D. Xu, and L. Hu (2023, First quarter). A survey on metaverse: Fundamentals, security, and privacy. *IEEE Communications Surveys and Tutorials 25*(1), 319–352.

Welsh, D. J. and M. B. Powell (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal 10*(1), 85–86.

Wu, W., N. Chen, C. Zhou, M. Li, X. Shen, W. Zhuang, and X. Li (2021). Dynamic ran slicing for service-oriented vehicular networks via constrained learning. *IEEE Journal on Selected Areas in Communications 39*(7), 2076–2089.

Xiang, H., S. Yan, and M. Peng (2020). A realization of fog-ran slicing via deep reinforcement learning. *IEEE Transactions on Wireless Communications 19*(4), 2515–2527.

Yan, Y., Q. Hu, and D. M. Blough (2021). Feasibility of multipath construction in mmwave backhaul. In *Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 81–90.

Yang, Z., Q. Zhang, and Z. Niu (2012). Throughput improvement by joint relay selection and link scheduling in relay-assisted cellular networks. *IEEE Transactions on Vehicular Technology 61*(6), 2824–2835.

Ye, H., G. Y. Li, and B.-H. Juang (2017). Power of deep learning for channel estimation and signal detection in OFDM systems. *IEEE Wireless Communications Letters 7*(1), 114–117.

Ye, H., G. Y. Li, and B.-H. Juang (2021). Deep learning based end-to-end wireless communication systems without pilots. *IEEE Transactions on Cognitive Communications and Networking 7*(3), 702–714.

Ye, H., G. Y. Li, and B.-H. F. Juang (2019). Deep reinforcement learning based resource allocation for v2v communications. *IEEE Transactions on Vehicular Technology 68*(4), 3163–3173.

Ye, H., L. Liang, G. Y. Li, and B.-H. Juang (2020). Deep learning-based end-to-end wireless communication systems with conditional gans as unknown channels. *IEEE Transactions on Wireless Communiations 19*(5), 3133–3143.

Ye, R., Z. Ni, F. Wu, S. Chen, and Y. Wang (2023). Personalized federated learning with inferred collaboration graphs. In *Proceedings of International Conference on Machine Learning.*

Yeniay, O. (2005). Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications 10*(1), 45–56.

Yu, Y., E. Dutkiewicz, X. Huang, and M. Mueck (2013). Downlink resource allocation for next generation wireless networks with inter-cell interference. *IEEE Transactions on Wireless Communications 12*(4), 1783–1793.

Zantedeschi, V., A. Bellet, and M. Tommasi (2020). Fully decentralized joint learning of personalized models and collaboration graphs. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pp. 864–874.

Zanzi, L., V. Sciancalepore, A. Garcia-Saavedra, H. D. Schotten, and X. Costa-Pérez (2021). Laco: A latency-driven network slicing orchestration in beyond-5G networks. *IEEE Transactions on Wireless Communications 20*(1), 667–682.

Zhang, C., P. Patras, and H. Haddadi (2019). Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys and tutorials 21*(3), 2224–2287.

Zhang, H., N. Liu, X. Chu, K. Long, A.-H. Aghvami, and V. C. Leung (2017). Network slicing based 5G and future mobile networks: mobility, resource management, and challenges. *IEEE communications magazine 55*(8), 138–145.

Zhang, J., S. Chen, X. Wang, and Y. Zhu (2021). DeepReserve: Dynamic edge server reservation for connected vehicles with deep reinforcement learning. In *Proceedings of IEEE Conference on Computer Communications*, pp. 1–10.

Zhang, Z., T. Ji, H. Shi, C. Li, Y. Huang, and L. Yang (2023). A self-supervised learning-based channel estimation for IRS-aided communication without ground truth. *IEEE Transactions on Wireless Communications 22*(8), 5446–5460.

Zhao, N., Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang (2019). Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks. *IEEE Transactions on Wireless Communications 18*(11), 5141–5152.

Zhao, Y., M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra (2018). Federated learning with non-IID data. *arXiv preprint arXiv:1806.00582*.

Zhao, Z., M. C. Vuran, F. Guo, and S. D. Scott (2021). Deep-waveform: A learned OFDM receiver based on deep complex-valued convolutional networks. *IEEE Journal on Selected Areas in Communications 39*(8), 2407–2420.

Zhou, L., X. Hu, E. C.-H. Ngai, H. Zhao, S. Wang, J. Wei, and V. C. M. Leung (2016). A dynamic graph-based scheduling and interference coordination approach in heterogeneous cellular networks. *IEEE Transactions on Vehicular Technology 65*(5), 3735–3748.

Zhu, H., V. Gupta, S. S. Ahuja, Y. Tian, Y. Zhang, and X. Jin (2021). Network planning with deep reinforcement learning. In *Proceedings of ACM Special Interest Group on Data Communication*, pp. 258–271.

20001372

20011372

Appendix A

Proof of Theorems in Chapter 2

Proof of Theorem 2.1

First, consider one clique k on the first (root) level of bottleneck structure that has no predecessor flow. The flow rates in this clique are the same and can be computed by Eq. (2.3) in Algorithm 1, which is

$$s_k = r_f = \frac{1}{\sum_{l \in \mathcal{L}_k} \frac{b_l}{C_l}}, \forall f \in \mathcal{F}_l, \tag{A.1}$$

where $b_l = |\mathcal{F}_l^{\mathrm{u}}| = |\mathcal{F}_l|$. By using its equivalent capacity, these flow rates can also be computed by allocating the clique equivalent capacity among all the unconverged flows, which is

$$r_f = \frac{R_k}{\sum_{l \in \mathcal{L}_k} b_l} = \frac{\sum_{l \in \mathcal{L}_k} p_{k,l} C_l}{\sum_{l \in \mathcal{L}_k} b_l}, \tag{A.2}$$

where the clique equivalent throughput R_k is referred to Definition 2.3. Based on the max-min rate allocation in Algorithm 1, the unconverged flows are allocated with equal resource shares, that is,

$$\frac{p_{k,1}}{b_1} = \frac{p_{k,2}}{b_2} = \ldots = \frac{p_{k,l}}{b_l} = \beta, \tag{A.3}$$

where b_l is an positive integer and β is the positive ratio of resource share. The case of $b_l = 0$ is naturally excluded here, since $p_{k,l} = 0$ if $b_l = 0$. Considering Eq. (A.1), (A.2), and (A.3), the virtual scheduling variables can be derived as

$$p_{k,l} = \frac{b_l}{\sum_{l \in \mathcal{L}_k} \frac{b_l}{C_l}} \frac{\sum_{l \in \mathcal{L}_k} b_l}{\sum_{l \in \mathcal{L}_k} b_l C_l}, \forall l \in \mathcal{L}_k. \tag{A.4}$$

Specially, if all the links in clique k have the same capacity, then Eq. (A.4) is simplified as $p_{k,l} = \frac{b_l}{\sum_{l \in \mathcal{L}_k} b_l}$, and the clique equivalent capacity in this case is a convex combination of link capacities.

In the following, we prove that the uniquely derived $p_{k,l}$ satisfies the constraints in Definition 2.3. Obviously, $p_{k,l}$ is non-negative. Based on Eq. (A.3) and (A.4), the problem of proving $\sum_{l \in \mathcal{L}_k} p_{k,l} = \beta \sum_{l \in \mathcal{L}_k} b_l \leq 1$ is transformed into proving the following inequality:

$$\frac{1}{\sum_{l\in\mathcal{L}_k}\frac{b_l}{C_l}}\frac{\sum_{l\in\mathcal{L}_k}b_l}{\sum_{l\in\mathcal{L}_k}b_lC_l}\leq\frac{1}{\sum_{l\in\mathcal{L}_k}b_l}. \tag{A.5}$$

The above inequality is reformed as

$$\left(\sum_{l\in\mathcal{L}_k}b_l\right)^2\leq\sum_{l\in\mathcal{L}_k}\frac{b_l}{C_l}\sum_{l\in\mathcal{L}_k}b_lC_l, \tag{A.6}$$

and further as

$$\sum_{i\in\mathcal{L}_k}\sum_{j\in\mathcal{L}_k,j\neq i}2b_ib_j\leq\sum_{i\in\mathcal{L}_k}\sum_{j\in\mathcal{L}_k,j\neq i}(\frac{C_i}{C_j}+\frac{C_j}{C_i})b_ib_j. \tag{A.7}$$

Since we have $C_i/C_j+C_j/C_i\geq 2,\forall C_i,C_j\geq 0$, the inequality (A.7) always holds true and the equality is obtained with $C_i=C_j,\forall i,j\in\mathcal{L}_k$. Therefore, the uniquely derived virtual scheduling variables $\{p_{k,l},\forall l\in\mathcal{L}_k\}$ satisfy the constraints in Definition 2.3. With this, the proof for the cliques on the root level is completed.

Next, consider one clique k on the remaining levels of the bottleneck structure that has a set of predecessor flows with the same converged rates r. Let $a_l=|\mathcal{F}_l^{\mathrm{c}}|$, $d_l=b_l-a_l=|\mathcal{F}_l^{\mathrm{u}}|$, and the flow rates for the unconverged flows is derived by in Algorithm 1 as

$$s_k=\frac{1-\sum_{l\in\mathcal{L}_k}\frac{a_lr}{C_l}}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}}. \tag{A.8}$$

Similarly, they can also be computed by allocating the remaining clique equivalent capacity among the unconverged flows, which is

$$s_k=\frac{R_k-\sum_{l\in\mathcal{L}_k}p_{k,l}a_lr}{\sum_{l\in\mathcal{L}_k}d_l}=\frac{\sum_{l\in\mathcal{L}_k}p_{k,l}(C_l-a_lr)}{\sum_{l\in\mathcal{L}_k}d_l}. \tag{A.9}$$

Based on the max-min rate allocation in Algorithm 1, the unconverged flows are allocated with equal resource share, which follows

$$\frac{p_{k,1}}{d_1}=\frac{p_{k,2}}{d_2}=...=\frac{p_{k,l}}{d_l}=\theta, \tag{A.10}$$

where d_l and θ are positive numbers. Considering Eq. (A.8), (A.9), and (A.10), the virtual scheduling variables can be derived as

$$p_{k,l}=\frac{d_l\left(1-\sum_{l\in\mathcal{L}_k}\frac{a_lr}{C_l}\right)}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}}\frac{\sum_{l\in\mathcal{L}_k}d_l}{\sum_{l\in\mathcal{L}_k}d_l(C_l-a_lr)} \tag{A.11}$$

Similar to the derivation for the first case, we prove that the uniquely derived $p_{k,l}$ satisfies the constraints in Definition 2.3. Obviously, $p_{k,l}$ is non-negative. The problem of proving $\sum_{l\in\mathcal{L}_k}p_{k,l}=\theta\sum_{l\in\mathcal{L}_k}d_l\leq 1$ is transformed into proving the following inequality:

$$\begin{aligned}\theta&=\frac{1-\sum_{l\in\mathcal{L}_k}\frac{a_lr}{C_l}}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}}\frac{\sum_{l\in\mathcal{L}_k}d_l}{\sum_{l\in\mathcal{L}_k}d_l(C_l-a_lr)}\\&\leq\frac{1}{\sum_{l\in\mathcal{L}_k}d_l}.\end{aligned} \tag{A.12}$$

20001372

To prove the above inequality, the following transformation is conducted:

$$\begin{aligned}
\theta &= \frac{\sum_{l\in\mathcal{L}_k} d_l[1 - r\sum_{l\in\mathcal{L}_k}\frac{a_l}{C_l}]}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_lC_l - r\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_la_l} \\
&= \frac{\sum_{l\in\mathcal{L}_k} d_l[1 - r\sum_{l\in\mathcal{L}_k}\frac{a_l}{C_l}]}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_lC_l[1 - \frac{r\sum_{l\in\mathcal{L}_k} d_la_l}{\sum_{l\in\mathcal{L}_k} d_lC_l}]} \\
&\le \frac{\sum_{l\in\mathcal{L}_k} d_l}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_lC_l} \\
&\le \frac{1}{\sum_{l\in\mathcal{L}_k} d_l}.
\end{aligned} \tag{A.13}$$

With this, we prove that $\{p_{k,l}, \forall l \in \mathcal{L}_k\}$ satisfies the constraints in Definition 2.3.

Finally, consider one clique k that has a set of predecessor flows with various converged rates $\hat{r}_m, m = 1, 2, ..., M$ in the bottleneck structure. Let $a_{l,m} = \mathcal{F}^c_{l,m}, d_l = |\mathcal{F}_l| - \sum_{m=1}^{M} |\mathcal{F}^c_{l,m}|$. Following the similar methodology of combining two ways of flow rate computation, the virtual scheduling variables can be derived as

$$p_{k,l} = \frac{d_l\sum_{l\in\mathcal{L}_k} d_l[1 - \sum_{m=1}^{M}\hat{r}_m\sum_{l\in\mathcal{L}_k}\frac{a_{l,m}}{C_l}]}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_lC_l[1 - \sum_{m=1}^{M}\frac{\hat{r}_m\sum_{l\in\mathcal{L}_k} d_la_{l,m}}{\sum_{l\in\mathcal{L}_k} d_lC_l}]}, \tag{A.14}$$

and

$$\frac{p_{k,1}}{d_1} = \frac{p_{k,2}}{d_2} = ... = \frac{p_{k,l}}{d_l} = \theta, \tag{A.15}$$

where d_l and θ are positive. Similar to the previous case, to prove $\sum_{l\in\mathcal{L}_k} p_{k,l} = \theta\sum_{l\in\mathcal{L}_k} d_l \le 1$, the key inequality below is proved as follows:

$$\begin{aligned}
\theta &= \frac{\sum_{l\in\mathcal{L}_k} d_l[1 - \sum_{m=1}^{M}\hat{r}_m\sum_{l\in\mathcal{L}_k}\frac{a_{l,m}}{C_l}]}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_lC_l[1 - \sum_{m=1}^{M}\frac{\hat{r}_m\sum_{l\in\mathcal{L}_k} d_la_{l,m}}{\sum_{l\in\mathcal{L}_k} d_lC_l}]} \\
&\le \frac{\sum_{l\in\mathcal{L}_k} d_l}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_lC_l} \\
&\le \frac{1}{\sum_{l\in\mathcal{L}_k} d_l}.
\end{aligned} \tag{A.16}$$

Therefore, the uniquely derived virtual scheduling variables satisfy the constraints in Definition 2.3, and thus Theorem 2.1 is proved to be true.

20011372

Proof of Theorem 2.2

The clique perturbation is obtained by imposing an infinitely small perturbation δ on each of clique j's link capacities, i.e., $\Delta R_j = \sum_{l \in \mathcal{L}_k} p_{k,l}\delta$. To prove Eq. (2.7) and (2.8), it is equivalent to prove

$$\tilde{s}_k = s_k + \Delta s_k(\delta \sum_{l \in \mathcal{L}_j} p_{j,l}), \forall k \in \hat{\mathcal{Q}}, \tag{A.17}$$

$$\tilde{r}_f = r_f + \Delta r_f(\delta \sum_{l \in \mathcal{L}_j} p_{j,l}), \forall f \in \mathcal{F}, \tag{A.18}$$

where $\tilde{s}_k$, $\tilde{r}_f$ are new values after perturbation.

An induction approach applied in the wired network (Ros-Giralt et al., 2022) is followed to prove Eq. (A.18) and (A.17). Similar to Definition 5 in Ros-Giralt et al. (2022), clique x or flow f is said to be in the clique k's influence region, provided that the clique fair share s_x of clique x or the flow rate r_f can be impacted by the change of the clique equivalent throughput R_k of clique k. Let $\mathcal{I}(j)$ be the influence region set. Consider any vertex $y \in \mathcal{V}_b$ in the constructed bottleneck structure, where it can be a flow or a clique. If it is not in the influence region of clique j, i.e., $y \notin \mathcal{I}(j)$, it is evident that $\Delta r_y = 0, \forall y \in \mathcal{F}$ or $\Delta s_y = 0, \forall y \in \hat{\mathcal{Q}}$. Thus, Eq. (A.18) and (A.17) hold true. For any vertex $y \in \mathcal{I}(j)$, the following proof is given by induction.

To start with, consider vertex y as the the root node in $\mathcal{I}(j)$. Vertex y must be a clique and it has no predecessor vertex. Hence, we replace the notation y by k. The only impact on clique k's fair share must directly come from the perturbation source clique j. Let $\mathcal{M}_{k,j} = \mathcal{L}_k \cap \mathcal{L}_j$, representing the common links of two cliques. Then the new fair share value for clique k can be derived as follows:

$$\begin{aligned}
\tilde{s}_k = \frac{\tilde{R}_k}{|\mathcal{B}_k|} &= \frac{R_k + \Delta R_k}{|\mathcal{B}_k|} \\
&= s_k + \frac{\delta \sum_{l \in \mathcal{M}_{k,j}} p_{k,l}}{|\mathcal{B}_k|} \\
&= s_k + \frac{\sum_{l \in \mathcal{M}_{k,j}} p_{k,l} / \sum_{l \in \mathcal{L}_j} p_{j,l}}{|\mathcal{B}_k|} \delta \sum_{l \in \mathcal{L}_j} p_{j,l} \\
&= s_k + \Delta s_k(\delta \sum_{l \in \mathcal{L}_j} p_{j,l}),
\end{aligned} \tag{A.19}$$

where $\mathcal{B}_k$ represent the successor vertices of clique k in the bottleneck structure. The above shows Eq. (A.17) holds true.

Next, following the methodology of induction, we assume y is any vertex in $\mathcal{I}(j)$ except the root node, and each predecessor vertex of y in $\mathcal{I}(j)$ satisfies Eq. (A.18) if it is a flow, or Eq. (A.17) if it is a clique. In the following, it is proved that vertex y itself satisfies one of the two equations, which completes the induction.

Consider the first case that vertex y is a flow, i.e., $y \in \mathcal{F}$, and the index y is replaced by f. It is assumed by induction that the predecessor cliques of flow f satisfies Eq. (A.17). Combined with the propagation equation

20011372

(2.6), it is shown that:

$$
\begin{aligned}
\tilde{r}_f &= \min_{k\in\hat{\mathcal{P}}_f} \tilde{s}_k \\
&= \min_{k\in\hat{\mathcal{P}}_f} \left(s_k + \Delta s_k \delta \sum_{l\in\mathcal{L}_j} p_{j,l} \right) \\
&= r_f + \delta \sum_{l\in\mathcal{L}_j} p_{j,l} \left(\min_{k\in\mathcal{P}_f^w} \Delta s_k \right) \\
&= r_f + \Delta r_f \left(\delta \sum_{l\in\mathcal{L}_j} p_{j,l} \right),
\end{aligned}
\tag{A.20}
$$

where $\hat{\mathcal{P}}_f$ represents the predecessor vertices of flow f in the bottleneck structure. This states that Eq. (A.18) holds true.

Consider the second case that vertex y is a bottleneck clique, i.e., $y \in \hat{\mathcal{Q}}$, and thus the index y is replaced by k. Based on the max-min allocation of clique y's available throughput, we have

$$
\tilde{s}_k = \frac{\tilde{R}_k - \sum_{f\in\mathcal{P}_k} \sum_{l\in\mathcal{L}_{k,f}} p_{k,l}\tilde{r}_f}{|\mathcal{B}_k|},
\tag{A.21}
$$

where $\tilde{R}_k = R_k + \Delta R_k$, and $\mathcal{P}_k$ represents the predecessor flows of clique k in the bottleneck structure. Since $\tilde{r}_f$ satisfies Eq. (A.18) based on induction, we have $\tilde{r}_f = r_f + \Delta r_f(\delta \sum_{l\in\mathcal{L}_j} p_{j,l})$. Also, s_k is expressed as $s_k = \left(R_k - \sum_{f\in\mathcal{P}_k} \sum_{l\in\mathcal{L}_{k,f}} p_{k,l} r_f\right) / |\mathcal{B}_k|$. Therefore, Eq. (A.21) is further transformed into

$$
\begin{aligned}
\tilde{s}_k &= s_k + \frac{\Delta R_k - \sum_{f\in\mathcal{P}_k} \sum_{l\in\mathcal{L}_k\cap\mathcal{R}_f} p_{k,l}\Delta r_f \Delta R_j}{|\mathcal{B}_k|} \\
&= s_k + \frac{\Delta R_k/\Delta R_j - \sum_{f\in\mathcal{P}_k} \sum_{l\in\mathcal{L}_k\cap\mathcal{R}_f} p_{k,l}\Delta r_f}{|\mathcal{B}_k|} \Delta R_j \\
&= s_k + \Delta s_k (\delta \sum_{l\in\mathcal{L}_j} p_{j,l}),
\end{aligned}
\tag{A.22}
$$

which states that Eq. (A.18) holds true. Note that based on the proof in Ros-Giralt et al. (2022), it suffices to use the original bottleneck structure with the imposed perturbation.

To this end, for any vertex in the bottleneck structure, either Eq. (2.7) or Eq. (2.8) is proved to hold true. The algorithm output of *CliqueGrad* is $\hat{\mathbf{g}}(j) = \sum_{f\in\mathcal{F}} \Delta r_f = \sum_{f\in\mathcal{F}} \frac{\partial r_f}{\partial R_j}$, which is exactly the clique gradient $\mathbf{g}(j)$. With that, the proof of Theorem 2.2 is completed.

20011372

20011372

Appendix B

Proof of Theorems in Chapter 3

Proof of Theorem 3.1

It is shown in (Yeniay, 2005) that problems P1 and P2 must be equivalent when the coefficients of the penalty function go to infinity. The optimal objective values of P1 and P2 are denoted by $f(\mathbf{x}^*)$ and $g(\mathbf{y}^*)$ respectively, where $\mathbf{x}^*$ and $\mathbf{y}^*$ are the optimal solutions. Let $\mathcal{F}$ and $\mathcal{H}$ denote the feasible region of P1 and P2. Obviously, $\mathbf{x}^* \in \mathcal{F}$ leads to $\mathbf{x}^* \in \mathcal{H}$. When $\mathbf{y}^* \in \mathcal{H}$ and $\mathbf{y}^* \in \mathcal{F}$ are simultaneously satisfied, problems P1 and P2 have the same optimal solution $\mathbf{x}^* = \mathbf{y}^*$.

There exists an upper bound for the optimal value $f(\mathbf{x}^*)$, which is the total number of PRB resources in multiple cells BN (B is the number of cells, and N is the number of PRBs in one radio frame).

When $\mathbf{y}^*$ falls out of the feasible region $\mathcal{F}$, the penalty term $\sum_{b\in\mathcal{B}}\sum_{m\in\mathcal{M}} L_{b,m}$ must be at least one. Here, if λ takes on values larger than BN, it is guaranteed that $g(\mathbf{y}^*)$ must be larger than $f(\mathbf{x}^*)$. Since the reward is set to be the negative of objective values, the DRL agent always receives smaller rewards if the sampled actions fall out of the feasible region $\mathcal{F}$ of problem P1.

Proof of Theorem 3.2

If transmission rates $r_{u,i}^{(b,m)}$ are assumed to be uniformly distributed over all the PRBs and the constraint (3.14) is removed in P3, graph coloring that optimizes the chromatic number can be reduced to problem P3. Since it is NP-hard to compute the chromatic number, problem P3 is proven to be NP-hard.

Proof of Theorem 3.3

Let B denote the number of BSs, U denote the number of users, and N denote the number of PRBs. For the following analysis, we assume that each BS has $\frac{U}{B}$ users.

The complexity of Welsh-Powell algorithm can be divided into two parts: calculate the degree of nodes $O(B\log B)$ and sort nodes by degrees $O(N^2)$. The masking mechanism is checking the neighbors of each cell-edge user, which is described by the complexity $O(\frac{U^2N}{B})$. For the greedy allocation part, the priority order of users can be obtained in $O(\frac{U}{B}\log\frac{U}{B})$. Then PRBs are sorted by rates and allocated sequentially, which has the complexity $O(U\log\frac{U}{B} + UN\log N + BN)$. Therefore, the second stage algorithm has the order of complexity of $O(B\log B + N^2 + U\log\frac{U}{B} + UN\log N + BN)$.

20011372

20011372

Appendix C

Proof of Theorem in Chapter 5

Proof of Theorem 5.1

As stated in the main text, the proof of generalization bound requires two major components: 1) the upper bound of the difference between weighted expected risk $F_{\mathbf{w}_m}(\cdot)$ and single-domain expected risk $F_m(\cdot)$, i.e.,

$$|F_{\mathbf{w}_m}(h) - F_m(h)| \leq \sum_{j=1}^{M} w_{mj} \left(\frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}} \left(\mathcal{D}_m, \mathcal{D}_j \right) + \lambda_{mj} \right), \forall h \in \mathcal{H}, \tag{C.1}$$

and 2) the probabilistic upper bound of the difference between empirical weighted risk $\hat{F}_{\mathbf{w}_m}(\cdot)$ and expected weighted risk $F_{\mathbf{w}_m}(\cdot)$, i.e., for any $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$\left| \hat{F}_{\mathbf{w}_m}(h) - F_{\mathbf{w}_m}(h) \right| \leq \sqrt{\sum_{j=1}^{M} \frac{w_{mj}^2}{p_j} \left(\frac{c \log(2N) - \log \delta}{2N} \right)}, \forall h \in \mathcal{H}. \tag{C.2}$$

20011372

With these two major components, the upper bound of $F_m(\hat{h}_{\mathbf{w}_m})$ can be derived as follows. For any $\delta \in (0,1)$, with probability at least $1-\delta$, there exists

$$
\begin{aligned}
F_m\left(\hat{h}_{\mathbf{w}_m}\right) &\le F_{\mathbf{w}_m}\left(\hat{h}_{\mathbf{w}_m}\right) + \sum_{j=1}^{M} w_{mj}\left(\frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}\left(\mathcal{D}_m, \mathcal{D}_j\right) + \lambda_{mj}\right) && \text{(C.3)}\\
&\le \hat{F}_{\mathbf{w}_m}\left(\hat{h}_{\mathbf{w}_m}\right) + + \sqrt{\sum_{j=1}^{M} \frac{w_{mj}^2}{p_j}\left(\frac{c\log(2N) - \log\delta}{2N}\right)} + \sum_{j=1}^{M} w_{mj}\left(\frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}\left(\mathcal{D}_m, \mathcal{D}_j\right) + \lambda_{mj}\right) && \text{(C.4)}\\
&\le \hat{F}_{\mathbf{w}_m}\left(h_m^*\right) + \sqrt{\sum_{j=1}^{M} \frac{w_{mj}^2}{p_j}\left(\frac{c\log(2N) - \log\delta}{2N}\right)} + \sum_{j=1}^{M} w_{mj}\left(\frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}\left(\mathcal{D}_m, \mathcal{D}_j\right) + \lambda_{mj}\right) && \text{(C.5)}\\
&\le F_{\mathbf{w}_m}\left(h_m^*\right) + 2\sqrt{\sum_{j=1}^{M} \frac{w_{mj}^2}{p_j}\left(\frac{c\log(2N) - \log\delta}{2N}\right)} + \sum_{j=1}^{M} w_{mj}\left(\frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}\left(\mathcal{D}_m, \mathcal{D}_j\right) + \lambda_{mj}\right) && \text{(C.6)}\\
&\le F_m\left(h_m^*\right) + 2\sqrt{\sum_{j=1}^{M} \frac{w_{mj}^2}{p_j}\left(\frac{c\log(2N) - \log\delta}{2N}\right)} + \sum_{j=1}^{M} w_{mj}\left(d_{\mathcal{H}\Delta\mathcal{H}}\left(\mathcal{D}_m, \mathcal{D}_j\right) + 2\lambda_{mj}\right), && \text{(C.7)}
\end{aligned}
$$

Note that in the above proof, we have $\hat{F}_{\mathbf{w}_m}\left(h_m^*\right) \le F_{\mathbf{w}_m}\left(h_m^*\right)$. This is because $\hat{h}_{\mathbf{w}_m}$ is the optimal minimizer of $\hat{F}_{\mathbf{w}_m}(h)$, that is $\hat{h}_{\mathbf{w}_m} = \arg\min_{h\in\mathcal{H}} \hat{F}_{\mathbf{w}_m}(h)$.

20011372

Publication List

1. T. Wang, S. Chen, Y. Zhu, A. Tang, X. Wang (2022). LinkSlice: Fine-Grained Network Slice Enforcement Based on Deep Reinforcement Learning. *IEEE Journal on Selected Areas in Communications 40*(8), 2378–2394.
2. T. Wang, X. Wang (2024). DeepRP: Bottleneck Theory Guided Relay Placement for 6G Mesh Backhaul Augmentation. *IEEE Transactions on Mobile Computing 24*(3), 1744-1758.
3. T. Wang, X. Wang, Y.B. Lin (2024). SideSeeker: Contention-Based Distributed Relay Finding for Sidelink Mesh Networks. *IEEE Wireless Communications Letters 13*(10), 2802–2806.
4. T. Wang, X. Wang, G.Y. Li (2025). GraphRx: Graph-Based Collaborative Learning among Multiple Cells for Uplink Neural Receivers. *IEEE Conference on Computer Communications (INFOCOM).*
5. S. Wang, T. Wang, X. Wang (2025). FedPDA: Collaborative Learning for Reducing Online-Adaptation Frequency of Neural Receivers. *IEEE Conference on Computer Communications (INFOCOM).*
6. T. Wang, S. Wang, X. Wang, G.Y. Li (2024). Collaborative Learning for Less Online Retraining of Neural Receivers. *IEEE Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1-6.
7. T. Wang, X. Wang (2023). Boosting Capacity for 6G Terahertz Mesh Networks Based on Bottleneck Structures. *IEEE Global Communications Conference (GLOBECOM)*, pp. 4589–4594.
8. X. Wang, T.Wang, A. Tang, Z. Li (2024). Communication system and method. PCT/CN2023/085658.

20011372

20011372

Acknowledgments

Standing in front of McEwan Hall in Edinburgh, amidst the echoes of history embedded in the sandstone, I find myself reflecting on this long and meaningful doctoral journey. Time, often described as fleeting, has never felt swift to me in the past five years. Rather, it has stretched itself, with each moment engraved and each chapter vivid.

My Ph.D. journey began in the autumn of 2020 at Shanghai Jiao Tong University (SJTU)—a place forever rooted in my heart, with its vast green meadows, tranquil riversides, and our office building clad in deep red bricks. The years spent at SJTU were warmly shaped by invaluable mentorship and friendship from WangLab, led by my supervisor Prof. Xudong Wang, witnessing my foundational growth in both research abilities and personality. In the summer of 2023, my doctoral pursuit brought me to Imperial College London, nestled within the vibrant heart of London. With continuous support from ITP lab led by my co-supervisor Prof. Geoffrey Li, each step forward enriched my academic views and broadened my intellectual horizons. Finally, my path led me to Edinburgh, a city wrapped in mystery, history, and an almost magical charm. I met more intelligent minds with diversified backgrounds from NetSys group led by Prof. Mahesh Marina. Now I feel fortunate and honored to have gone so far in this journey, growing up from a hesitant college girl into a confident female academic.

I am truly honored to have been a recipient of SJTU Zhiyuan Honors Scholarship, which has not only provided me with the financial support but has also been a source of encouragement. The recognition bestowed upon me by this scholarship has inspired me to strive for excellence and contribute to the society. My deep gratitude goes to my mentors, colleagues, and friends at SJTU, Imperial College London, and The University of Edinburgh, whose support, kindness, and companionship transformed my inherently lonely journey into an unforgettable adventure filled with warmth and shared purpose.

In this moment of gratitude, I recognize first and foremost my main supervisor Prof. Wang. He has always demonstrated a strong sense of responsibility towards research and provided me with meticulous academic guidance. During times when I encountered difficulties, his guidance and support were instrumental in helping me move forward. I believe that his advice will resonate far beyond these years, shaping not only my research but also how I view and navigate the challenges of life. I am also very thankful to my co-supervisor Prof. Li and mentor Prof. Marina for their kind support and guidance while I was in the UK. Thank all the other defense committee members, Prof. Yibo Pi, Prof. Meixia Tao, Prof. Chong Han, and Prof. Honggang Zhang.

I also express my appreciation to all of my friends and colleagues. Thank Mengxin Yu, Bangzhao Zhai, and Cheng Huang for patiently guiding me into research when I was confused about my direction in my freshman year. Thank Dianhan Xie, Jiawei Zhang, and Xiaochen Zhou for always resolving my doubts promptly and enlightening our lives with humor and optimism. Thank Suhong Chen, Shuo Wang, Xin Wang, and Yuanzhe Huang for collaborating with me to deliver the projects. Thank Qimin Zhao and Chenhao Luo for those inspiring and pleasant talks we shared. Thank Yuhang Chen for giving me detailed guidance on Visa application and so on. Special thanks go to Yawen Chang and Lei Lei for offering me so much favor and comfort no matter whether I was in Shanghai or abroad. I could not have come this far without their support.

Thank Zijian Cao, Kaidi Xu, Yanzhen liu, and Ouya Wang for helping me settle down when I arrived in London. Thank Shan Sha and Fábio Coutinho for being my best friends in London. Thank María Gragera Garcés for the warm conversations and insightful tutorials. Thank Andrew Ferguson, Ujjwal Pawar, Leyang Xue, and Qingrui Pan for being so sincere and helpful to me in Edinburgh. Thank Prof. Paul Patras for offering me so much kindness as well as the best coffee in the UK.

Special thanks go to my brilliant friends at SJTU: Zhaohui Jiang, Yuanbo Li, Zhihan Zhang, Yufei Sheng, and Yinchen Ni. I will always miss the carefree time we spent together in Shanghai. I would like to specially thank my dearest friends from the Jinling high school, Huili Sun, Yilin Wu, Ningshu Yang, Rui Wang, and Xiaoyu Zhang for their massive support and encouragement throughout my growth. Even though we rarely live in the same city or even country these years, we are always there for each other.

I would like to extend my deepest gratitude to my partner, Mr. Zixiao Li, for his support and inspiration throughout my journey. The trust and bond we have built together are invaluable, and I am certain that this foundation will endure forever. Thank you, Zixiao, for being my greatest ally.

The deepest gratitude goes to my families especially my parents, whose unwavering love and support have transcended oceans, continents, and time zones. Without their unshakable faith in me, I could never have achieved all that I have, nor become the soul I am today. They have been my harbor, providing a safe and steady refuge where I, like a ship, could find solace and strength during turbulent times. For all of this and so much more, I am eternally grateful. Mom and Dad, you are my heart, my foundation, and my greatest blessing.

Lastly, I would like to say thank myself. Thank you for refusing to give up, even when the journey seemed endless; for standing as a warrior in the face of every obstacle; and for embracing growth, shedding tears, finding courage, and discovering moments of joy along the way.

20011372

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：2015年8月14日

上海交通大学

学位论文使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。

本学位论文属于 ☑**公开论文**

□**内部论文**，□1 年/□2 年/□3 年　解密后适用本授权书。

□**秘密论文**，____年（不超过 10 年）解密后适用本授权书。

□**机密论文**，____年（不超过 20 年）解密后适用本授权书。

（请在以上方框内打“√”）

学位论文作者签名：　　　　指导教师签名：

日期：2015年8月14日　　　　日期：2025年9月1日

20011372

020370910032

上海交通大学博士学位论文答辩决议书

姓　名	王天欣	学号	020370910032	所在学科	信息与通信工程
指导教师	王旭东	答辩日期	2025-08-14	答辩地点	龙宾楼403会议室
论文题目	基于AI算法在不同架构层提升无线网络吞吐量的研究				

投票表决结果：5/5/5（同意票数/实到委员数/应到委员数）　答辩结论：☑通过　□未通过

评语和决议：

本论文围绕“网络架构—资源管理—物理层”三个层面，面向提升无线网络吞吐量开展系统研究，具有重要的理论意义和应用价值。

作者系统归纳并评述了相关文献，扎实掌握了该领域国内外的研究现状和发展方向。论文的创新点如下：

1) 在网络架构层面，针对多跳无线回传网的中继节点部署问题，建立了基于图论的网络瓶颈理论，并在该理论指导下设计了一种基于强化学习的中继部署方案，大幅提升了回传网络的吞吐量；
2) 在资源管理层面，针对无线接入侧的多小区网络切片资源管理问题，设计了一种基于强化学习的细粒度无线资源分配方法，在保证链路级别软隔离、满足用户服务需求并遵循长期切片策略约束的同时，显著提升了接入网的吞吐量；
3) 在物理层的信道估计方面，针对基于深度学习的在线无标签训练问题，提出在信道估计任务基础上引入自监督任务，通过在线训练该自监督任务显著降低信道估计误差，从而提升链路速率；
4) 在物理层的接收机方面，针对基于深度学习的多小区协同训练问题，提出了基于合作图的个性化联邦学习方案，联合优化合作图权重以及各小区接收机的模型参数，显著降低了接收机误码率。

论文结构完整，叙述清晰，分析合理，数据翔实，结论明确，表明作者具有坚实宽广的基础理论和系统深入的专业知识，具备独立从事科学研究工作的能力。

在答辩过程中，作者表述清晰有条理，回答问题正确。经论文答辩委员会无记名投票，一致同意通过王天欣同学的博士学位论文答辩，建议校学位评定委员会授予其工学博士学位，并推荐参与优秀博士论文评选。

皮宜博

2025年 8 月 14日

答辩委员会成员签名	职务	姓名	职称	单位	签名
	主席	皮宜博	副教授	上海交通大学	皮宜博
	委员	李烨	教授	帝国理工学院	李烨
	委员	韩充	副教授	上海交通大学	韩充
	委员	陶梅霞	教授	上海交通大学	陶梅霞
	委员	张宏纲	教授	澳门城市大学	張宏綱
	秘书	韩充	副教授	上海交通大学	韩充