

Graph-based Machine Learning for Wireless Communications

Santiago Segarra*, Ananthram Swami†, Zhongyuan Zhao*

*Rice University, †US Army's DEVCOM Army Research Laboratory



RICE



Slides PDF



October 28th, 2024

28 October–1 November 2024 // Washington, DC, USA // IEEE Military Communications Conference

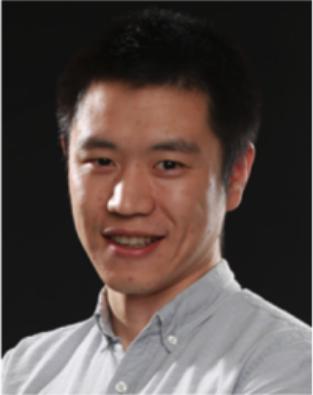
The team



Santiago Segarra
ECE, Rice



Ananthram Swami
DEVCOM, ARL



Zhongyuan Zhao
ECE, Rice



- ▶ Published 30+ papers in the area in the last 5 years

Acknowledgements

► Rice University

- ⇒ Arindam Chowdhury
- ⇒ Boning Li
- ⇒ Nicolas Zilberstein
- ⇒ Qing An
- ⇒ Ashu Sabharwal
- ⇒ Rahman Doost-Mohammady

► Army Research Lab

- ⇒ Gunjan Verma
- ⇒ Jake Perazzone
- ⇒ Kevin Chan
- ⇒ Chirag Rao

Key Takeaways

By 2:45 pm today, you will be able to answer the following questions:

- ▶ What are graph neural networks (GNNs)?
- ▶ Why are GNNs well suited to tackle problems in wireless communications?
- ▶ How have GNNs been applied to specific problems?
- ▶ What are open problems/challenges to which you can contribute?

Outline

Part I: Introduction to Graph Neural Networks

- a) Graph-based ML and GNNs
- b) Graphs, GNNs, and Wireless Networks

Part II: GNNs at the Physical Layer

- a) Introduction to issues at the physical layer
- b) Optimal Power Allocation & Beamforming: SISO and MIMO cases
- c) Optimal Power Allocation: Federated Learning
- d) Other Applications at the PHY layer

Part III: Graph-based ML for Wireless Networking

- a) Introduction to networking tasks
- b) Graph-based actor-critic reinforcement learning framework
- c) Fast & Differentiable Network Simulators

Conclusions and Future Directions

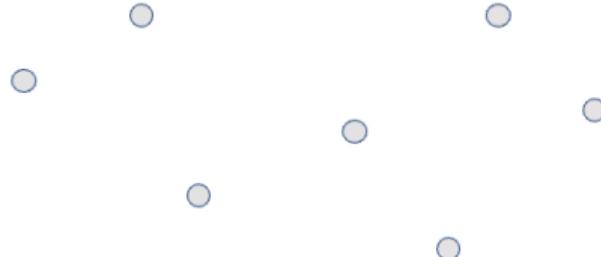
Part I: Introduction to Graph Neural Networks

Graph-based Machine Learning

Graph-based ML leverages the network structure of the data to improve learning and processing of these data

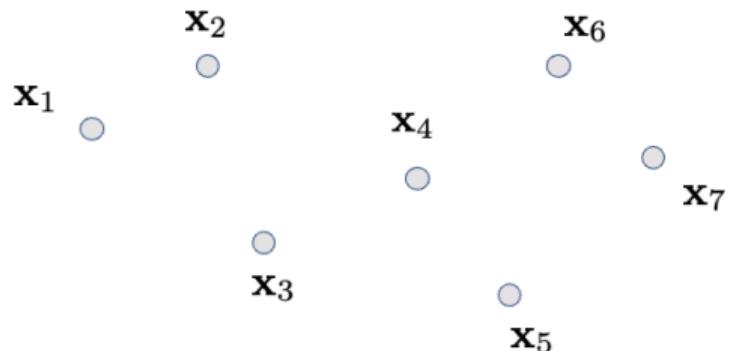
Graph-based Machine Learning

Graph-based ML leverages the network structure of the data to improve learning and processing of these data



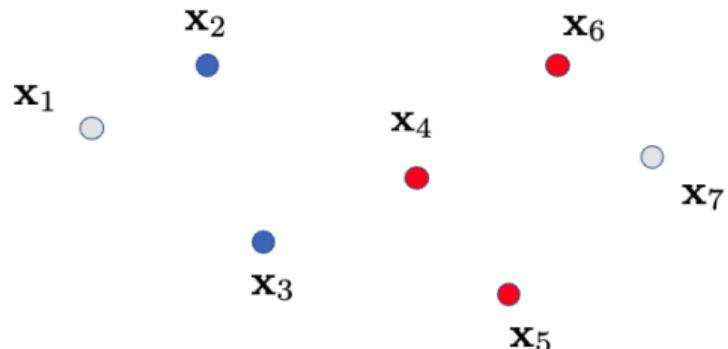
Graph-based Machine Learning

Graph-based ML leverages the network structure of the data to improve learning and processing of these data



Graph-based Machine Learning

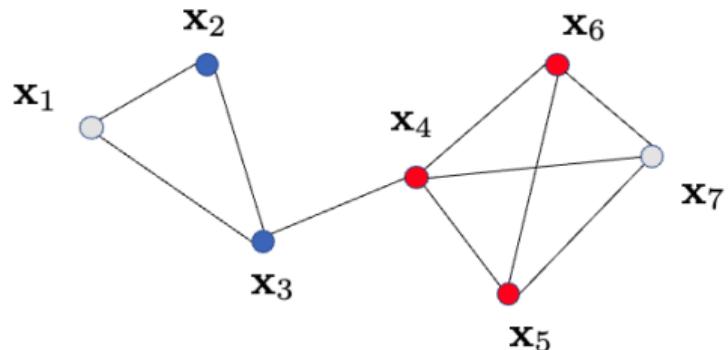
Graph-based ML leverages the network structure of the data to improve learning and processing of these data



- ▶ Classical **supervised learning** setting
⇒ Learn a **parametric** function that estimates the labels ⇒ $\hat{y}_i = f_{\theta}(\mathbf{x}_i)$

Graph-based Machine Learning

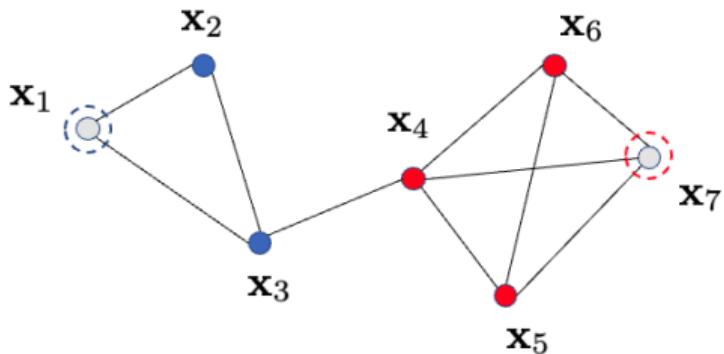
Graph-based ML leverages the network structure of the data to improve learning and processing of these data



- ▶ In some settings, relational structures between nodes are available
 - ⇒ Friendship in social networks or inhibition in protein networks
 - ⇒ Interference in comms networks

Graph-based Machine Learning

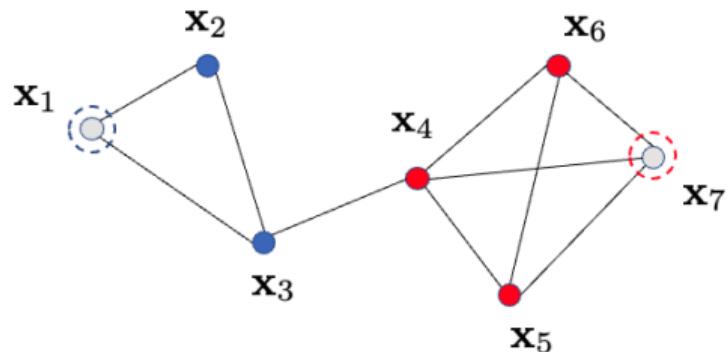
Graph-based ML leverages the network structure of the data to improve learning and processing of these data



- ▶ The structure also carries information about node labels
⇒ Estimate labels by combining both node features and graph structure

Graph-based Machine Learning

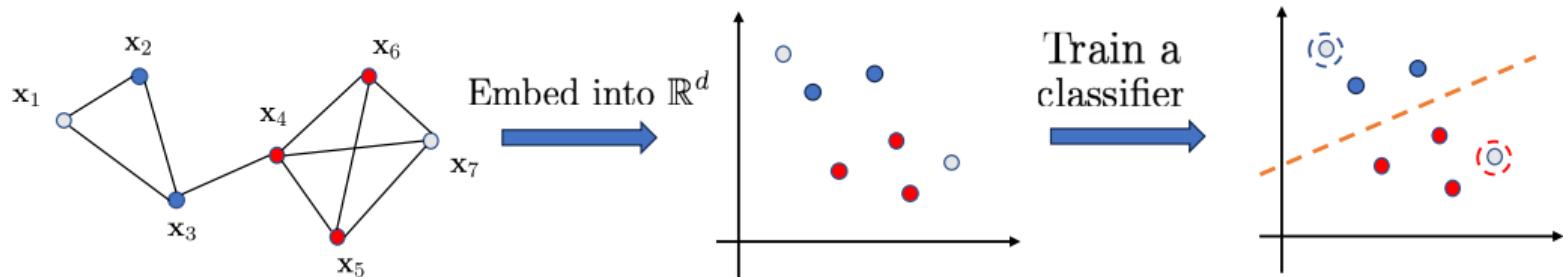
Graph-based ML leverages the network structure of the data to improve learning and processing of these data



$$\hat{y}_i = f_{\theta}(\{\mathbf{x}_j\}_{j=1}^N; \mathbf{A})$$

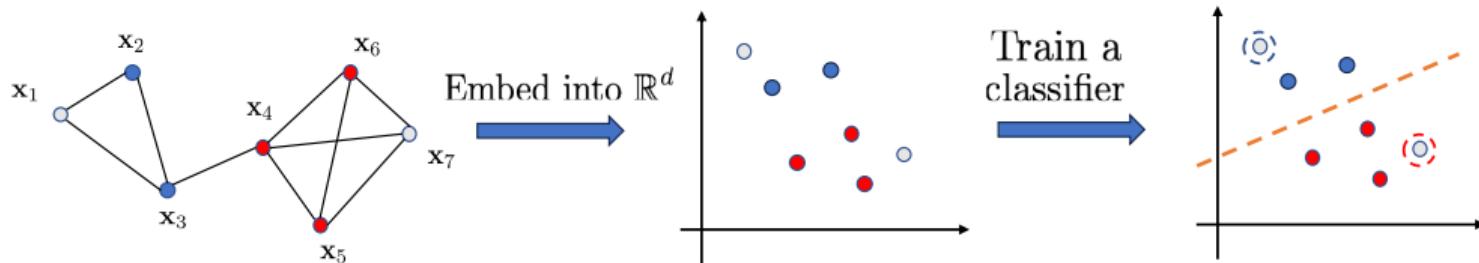
Graph Representation Learning

- Convert raw graph data into a low-dimensional vector representation



Graph Representation Learning

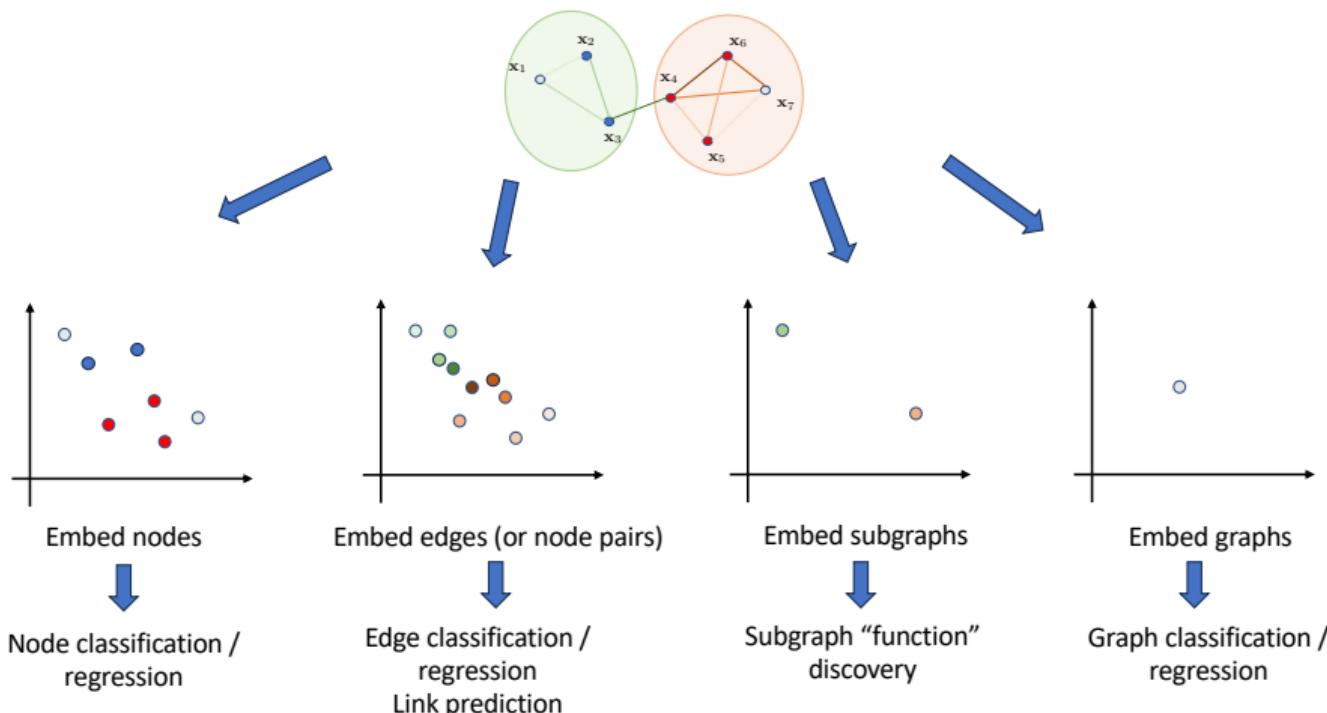
- Convert raw graph data into a low-dimensional vector representation



- Once in \mathbb{R}^d , we can apply the whole ML machinery
- Embedding can be **unsupervised**
⇒ “Closeness” in the graph is preserved as “closeness” in \mathbb{R}^d
- or **supervised**
⇒ Trained together with the downstream classifier
- We can embed other **graph elements beyond nodes**

Graph Representation Learning

- We can embed **nodes**, **edges**, **subgraphs**, and **whole graphs**



Graph Representation Learning

- ▶ “Shallow” embeddings (2014 - 2016): LINE, DeepWalk, node2vec
- ▶ $O(N)$ parameters are needed \Rightarrow No parameter sharing
- ▶ Inherently transductive \Rightarrow needs retraining for new nodes
- ▶ No node features \Rightarrow key in many applications

Graph Representation Learning

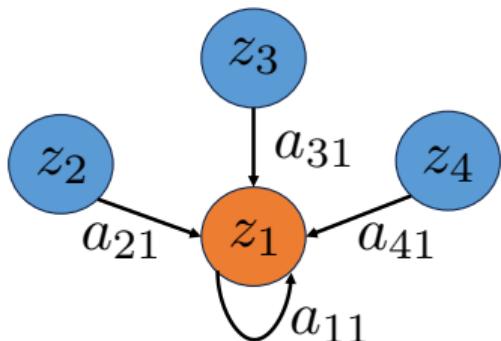
- ▶ “Shallow” embeddings (2014 - 2016): LINE, DeepWalk, node2vec
- ▶ $O(N)$ parameters are needed \Rightarrow No parameter sharing
- ▶ Inherently transductive \Rightarrow needs retraining for new nodes
- ▶ No node features \Rightarrow key in many applications

- ▶ “Deep” embeddings (2016 - present): GCN, GraphSAGE, and many others
- ▶ Graph neural networks address limitations of shallow embeddings

From GCNs to Message Passing Networks

- ▶ Every layer aggregates (one-hop) information and we stack several layers to increase the size of the “local” neighborhood influencing every node’s output

Graph Convolutional Network (GCN)



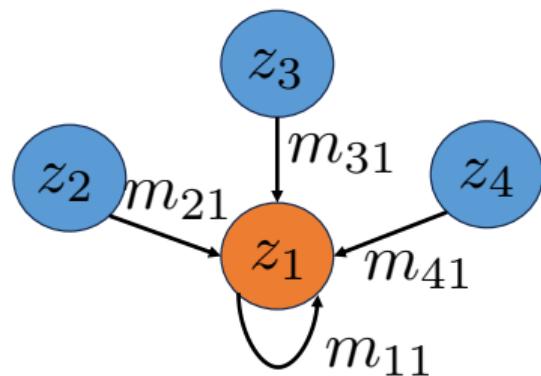
$$z_i^{(0)} = x_i$$

$$z_i^{(k+1)} = \sigma \left(\sum_{j \in (\mathcal{N}_i \cup \{i\})} w a_{ji} z_j^{(k)} \right)$$

From GCNs to Message Passing Networks

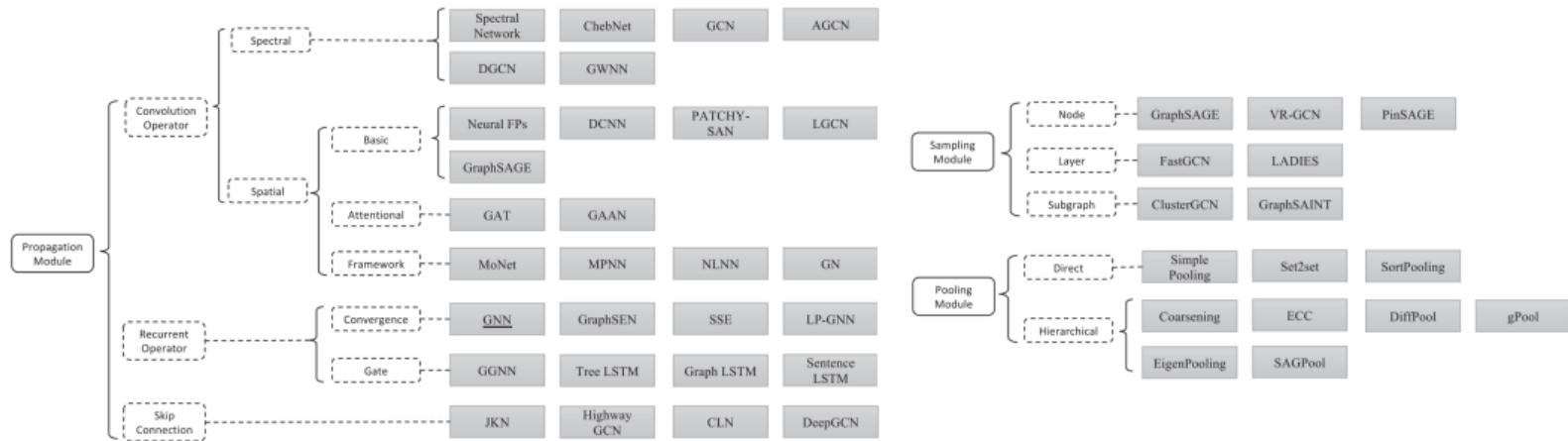
- Every layer aggregates (one-hop) information and we stack several layers to increase the size of the “local” neighborhood influencing every node’s output

Message-passing Neural Network (MPNN)



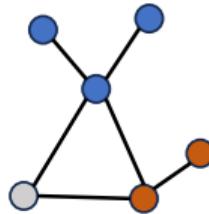
$$\begin{aligned} z_i^{(0)} &= x_i \\ m_{ji}^{(k)} &= f_e(z_i^{(k)}, z_j^{(k)}, e_{ji}) \\ z_i^{(k+1)} &= f_v \left(z_i^{(k)}, \sum_{j \in (\mathcal{N}_i \cup \{i\})} m_{ji}^{(k)} \right) \end{aligned}$$

A Zoo of GNNs has been developed

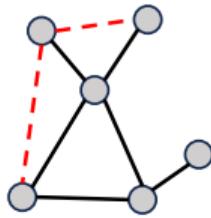


Credit: "Graph neural networks: A review of methods and applications"

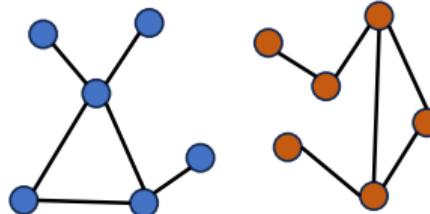
Typical generic problems tackled with GNNs



Node classification



Link prediction

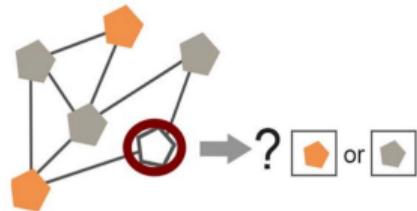


Graph classification

Translate into Domain-Specific Problems

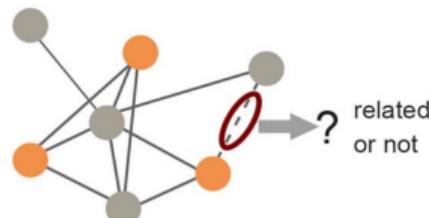
Node classification

-  Active protein function
-  Inactive protein function

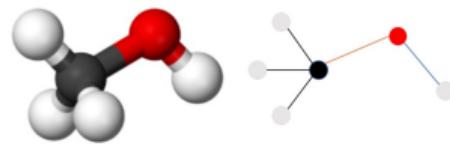


Link Prediction

-  RNA
-  Disease



Graph Classification

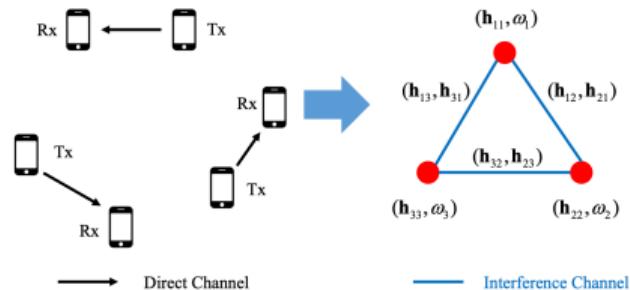


Soluble molecule or not

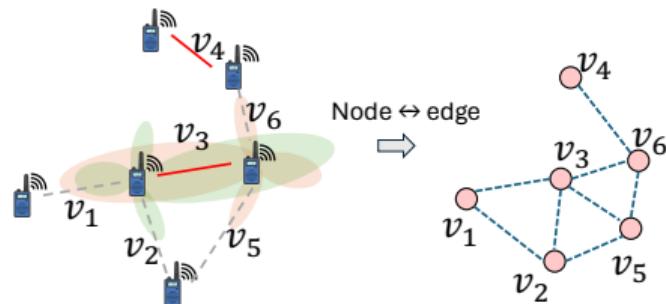
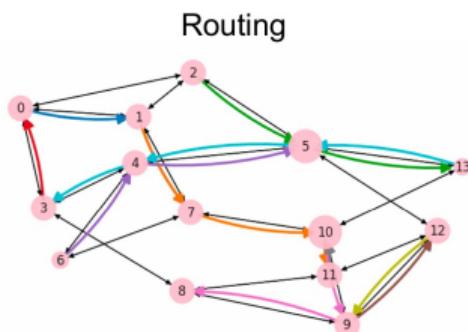
Credit: "Graph Neural Networks and their Current Applications in Bioinformatics", Zhang et al.

Credit: "Graph Neural Networks: A Review of Methods and Applications", Zhou et al.

Graphs in Wireless Communications



Credit: "Graph-based Deep Learning for Communication Networks: A Survey", Jiang



Why GNNs for Wireless Communications?

- ▶ Built-in **scalability**
 - ⇒ We can train and test with **different sizes of systems**

Why GNNs for Wireless Communications?

- ▶ Built-in **scalability**
 - ⇒ We can train and test with **different sizes of systems**
- ▶ Facilitate **distributed** implementation
 - ⇒ Forward-pass implementation based on **local computations**

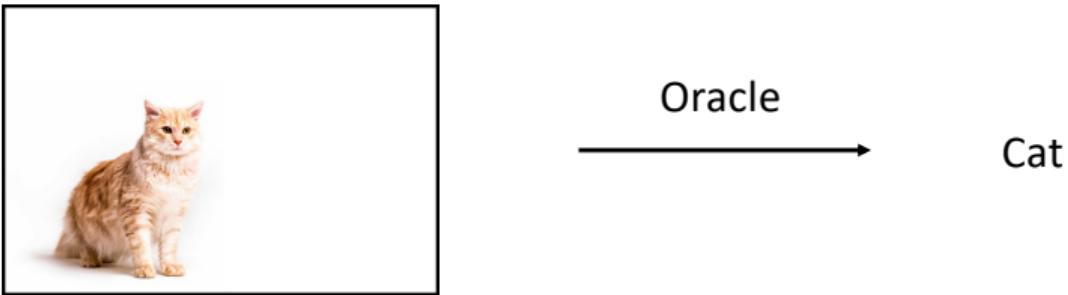
Why GNNs for Wireless Communications?

- ▶ Built-in **scalability**
 - ⇒ We can train and test with **different sizes of systems**
- ▶ Facilitate **distributed** implementation
 - ⇒ Forward-pass implementation based on **local computations**
- ▶ **Locality** plays a central role
 - ⇒ My optimal decision depends on the parts of the **network close to me**

Why GNNs for Wireless Communications?

- ▶ Built-in **scalability**
 - ⇒ We can train and test with **different sizes of systems**
- ▶ Facilitate **distributed** implementation
 - ⇒ Forward-pass implementation based on **local computations**
- ▶ **Locality** plays a central role
 - ⇒ My optimal decision depends on the parts of the **network close to me**
- ▶ Exploit the correct **symmetries**
 - ⇒ **Permutation equivariance/invariance** is a natural feature of network control

CNNs and translation invariance



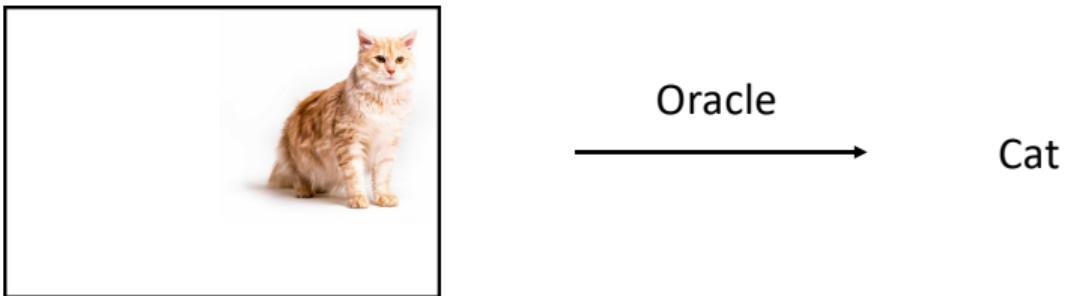
CNNs and translation invariance



Oracle

Cat

CNNs and translation invariance



- ▶ Architectures used for **object recognition** benefit from **translation invariance**
 ⇒ **Convolutional Neural Networks**
- ▶ Learning in the class of function to which the oracle belongs

GNNs and permutation equivariance

$$H \xrightarrow{\text{Oracle}} p^*$$

GNNs and permutation equivariance

$$\Pi \mathbf{H} \Pi^\top \xrightarrow{\text{Oracle}} \Pi \mathbf{p}^*$$

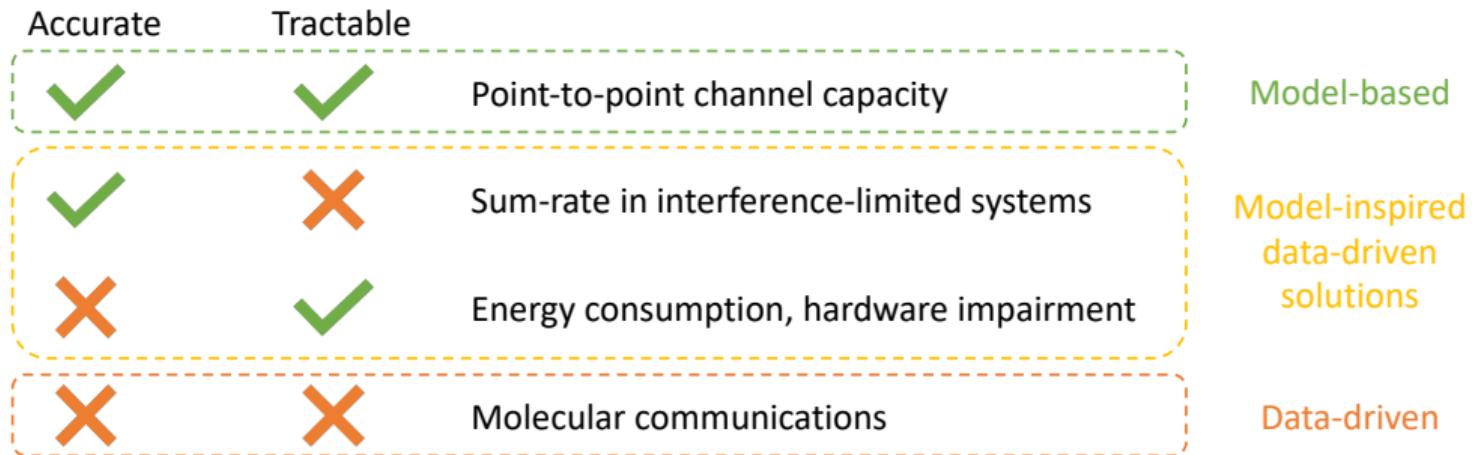
GNNs and permutation equivariance

$$\boldsymbol{\Pi} \mathbf{H} \boldsymbol{\Pi}^\top \xrightarrow{\text{Oracle}} \boldsymbol{\Pi} \mathbf{p}^*$$

- ▶ Architectures used for **power allocation** benefit from **permutation equivariance**
⇒ **Graph Neural Networks**
- ▶ Learning in the class of function to which the oracle belongs

Model-inspired Data-driven Solutions

Theoretical model



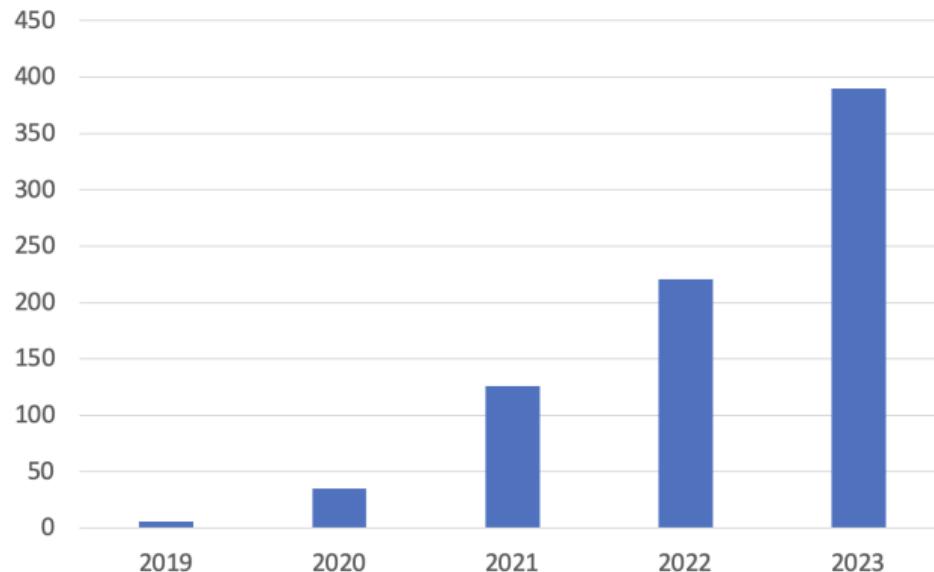
Credit: "Wireless Networks Design in the Era of Deep Learning: Model-Based, AI-Based, or Both?" Zappone et al., IEEE ToC, 2019

- ▶ Synergy between classical models and modern data-driven solutions

Overview of GNN applications to Wireless Comms

- Very dynamic field \Rightarrow Many new papers being published

Papers "GNN" + "Wireless Network"



Overview of GNN applications to Wireless Comms

- ▶ Very **dynamic field** ⇒ Many new papers being published
- ▶ **Several tutorials/surveys** in the area
 - ⇒ He et al., “An overview on the application of graph neural networks in wireless networks”, IEEE O. J. of the Comm. Soc., 2021
 - ⇒ Hu et al., “Distributed Machine Learning for Wireless Communication Networks: Techniques, Architectures, and Applications”, IEEE Comm. Surv. & Tut., 2021
 - ⇒ Shen et al., “Graph neural networks for wireless communications: From theory to practice”, IEEE Trans. Wireless Comm., 2022
 - ⇒ Lee et al., “Graph neural networks meet wireless communications: Motivation, applications, and future directions”, IEEE Wireless Comm., 2022

Overview of GNN applications to Wireless Comms

- ▶ Very **dynamic field** ⇒ Many new papers being published
- ▶ **Several tutorials/surveys** in the area
 - ⇒ Simeone, “A Very Brief Introduction to Machine Learning with Applications to Communication Systems”, IEEE Trans. on Cognitive Comm. and Netw., 2018
 - ⇒ Ahmad et al., “Machine Learning Meets Communication Networks: Current Trends and Future Challenges”, IEEE Access, 2020
 - ⇒ Ali et al., “6G White Paper on Machine Learning in Wireless Communication Networks”, Arxiv, 2020
 - ⇒ Jiang, “Graph-based deep learning for communication networks: A survey”, Computer Comm., 2022
 - ⇒ Suárez-Varela et al., “Graph Neural Networks for Communication Networks: Context, Use Cases and Opportunities”, IEEE Network, 2023

Overview of GNN applications to Wireless Comms

- ▶ Very dynamic field \Rightarrow Many new papers being published
- ▶ Several tutorials/surveys in the area
- ▶ A variety of problems have been tackled, including:
 - \Rightarrow Power allocation and beamforming
 - \Rightarrow Channel estimation
 - \Rightarrow Traffic prediction
 - \Rightarrow Spectrum allocation
 - \Rightarrow Cooperative caching
 - \Rightarrow Link scheduling
 - \Rightarrow Routing

Part II: GNNs at the Physical Layer

Overview

► Broad objective

- ⇒ Interference management in tactical wireless ad hoc networks
- ⇒ Network utility optimization under constraints

Overview

- ▶ **Broad objective**

- ⇒ Interference management in tactical wireless ad hoc networks
- ⇒ Network utility optimization under constraints

- ▶ **Domain-inspired learning and reusable models**

- ⇒ Combine classical algorithms with data-driven modules
- ⇒ Domain knowledge with neural acceleration

Overview

- ▶ **Broad objective**
 - ⇒ Interference management in tactical wireless ad hoc networks
 - ⇒ Network utility optimization under constraints
- ▶ **Domain-inspired learning and reusable models**
 - ⇒ Combine classical algorithms with data-driven modules
 - ⇒ Domain knowledge with neural acceleration
- ▶ **Learning under constraints**
 - ⇒ Near-optimal solution for the unconstrained problem
 - ⇒ Flexibility of learning to operate under multiple constraints
- ▶ **Intelligent system** leverages graph structure to allocate power
 - ⇒ Requires centralized training but deployment can be distributed

Motivation - Power Allocation

- ▶ Power and bandwidth are fundamental resources in communication
⇒ Key to determine the effective capacity of a wireless network

Motivation - Power Allocation

- ▶ Power and bandwidth are fundamental resources in communication
 - ⇒ Key to determine the effective capacity of a wireless network
- ▶ Randomly varying channel and user demand
 - ⇒ Optimal resource (re-)allocation essential for smooth functioning
- ▶ Algorithms must be robust against perturbations in the network

Motivation - Power Allocation

- ▶ Power and bandwidth are fundamental resources in communication
 - ⇒ Key to determine the effective capacity of a wireless network
- ▶ Randomly varying channel and user demand
 - ⇒ Optimal resource (re-)allocation essential for smooth functioning
- ▶ Algorithms must be robust against perturbations in the network
- ▶ We consider the optimal power allocation problem
 - ⇒ Fast, efficient, and robust solution

Optimal Power Allocation - SISO Case

Arindam Chowdhury *et al.*, IEEE ICASSP'21, IEEE ICASSP'22, IEEE Trans.
Wireless Communications 2021
Arindam Chowdhury, PhD thesis, Rice Univ. 2024

System Model

- ▶ Ad hoc network with m transmitter-receiver pairs
- ▶ Transmitter i has an associated receiver $r(i)$ for all $i \in \{1, m\}$

System Model

- ▶ Ad hoc network with m transmitter-receiver pairs
- ▶ Transmitter i has an associated receiver $r(i)$ for all $i \in \{1, m\}$
- ▶ Channel State Information (CSI) matrix $\mathbf{H}(t) \in \mathbb{R}^{m \times m}$
 - ⇒ Encodes (time-varying) channel characteristics
 - ⇒ $H_{ji}(t)$ represents the channel from Tx i to Rx $r(j)$ at time t

$$H_{ji}(t) = H_{ji}^P H_{ji}^F(t)$$

⇒ where $H_{ji}^P \propto \text{dist}(i, r(j))^{-k}$ and $H_{ji}^F(t) \sim \text{Rayleigh}(\alpha)$

System Model

- ▶ Ad hoc network with m transmitter-receiver pairs
- ▶ Transmitter i has an associated receiver $r(i)$ for all $i \in \{1, m\}$
- ▶ Channel State Information (CSI) matrix $\mathbf{H}(t) \in \mathbb{R}^{m \times m}$
 - ⇒ Encodes (time-varying) channel characteristics
 - ⇒ $H_{ji}(t)$ represents the channel from Tx i to Rx $r(j)$ at time t

$$H_{ji}(t) = H_{ji}^P H_{ji}^F(t)$$

⇒ where $H_{ji}^P \propto \text{dist}(i, r(j))^{-k}$ and $H_{ji}^F(t) \sim \text{Rayleigh}(\alpha)$

- ▶ Node State Information (NSI) matrix $\mathbf{X}(t) \in \mathbb{R}^{m \times d}$
 - ⇒ Encodes (time-varying) node features of the Tx-Rx pair
 - ⇒ # of packets that arrived, queue length, user priority

Problem Description

Given the CSI matrix $\mathbf{H}(t)$, the NSI matrix $\mathbf{X}(t)$, and a network utility function $u(\mathbf{H}(t), \mathbf{X}(t), \mathbf{p}(t))$, determine the optimal power allocation $\mathbf{p}(t) \in \mathbb{R}_+^m$

Problem Description

Given the CSI matrix $\mathbf{H}(t)$, the NSI matrix $\mathbf{X}(t)$, and a network utility function $u(\mathbf{H}(t), \mathbf{X}(t), \mathbf{p}(t))$, determine the optimal power allocation $\mathbf{p}(t) \in \mathbb{R}_+^m$

- ▶ Power constraint; Maximum power at each node $\Rightarrow p_i \leq p_{\max}$
- ▶ Network utility function: weighted sum rate across nodes $\sum_{i=1}^m \alpha_i c_i$
- ▶ Data rate at receiver i is given by (for noise variance σ^2)

$$c_i = \log_2 \left(1 + \frac{|H_{ii}|^2 p_i}{\sigma^2 + \sum_{j \neq i} |H_{ij}|^2 p_j} \right)$$

- ▶ Seeking a function $\mathbf{p}(\mathbf{H}, \mathbf{X})$ to optimize WSR

Classical Approach

- Weighted minimum mean-square error (WMMSE) [Shi *et al.*, TSP 2011]

⇒ Reformulate the optimization problem

$$\min_{\mathbf{w}, \mathbf{u}, \mathbf{v}} \sum_{i=1}^m (w_i e_i(\mathbf{H}, \mathbf{u}, \mathbf{v}) - \log w_i)$$

⇒ The optimal power p_i is given by v_i^2

Classical Approach

- Weighted minimum mean-square error (WMMSE) [Shi *et al.*, TSP 2011]

⇒ Reformulate the optimization problem

$$\min_{\mathbf{w}, \mathbf{u}, \mathbf{v}} \sum_{i=1}^m (w_i e_i(\mathbf{H}, \mathbf{u}, \mathbf{v}) - \log w_i)$$

⇒ The optimal power p_i is given by v_i^2

- WMMSE implemented as block coordinate descent, with closed-form iterations

$$u_i = \frac{H_{ii}v_i}{\sigma^2 + \sum_j H_{ji}^2 v_j^2}$$

$$w_i = \frac{1}{1 - u_i H_{ii} v_i}$$

$$v_i = \frac{\alpha_i u_i H_{ii} w_i}{\mu + \sum_j \alpha_j H_{ij}^2 u_j^2 w_j}$$

Classical Approach

- Weighted minimum mean-square error (WMMSE) [Shi *et al.*, TSP 2011]

⇒ Reformulate the optimization problem

$$\min_{\mathbf{w}, \mathbf{u}, \mathbf{v}} \sum_{i=1}^m (w_i e_i(\mathbf{H}, \mathbf{u}, \mathbf{v}) - \log w_i)$$

⇒ The optimal power p_i is given by v_i^2

- WMMSE implemented as block coordinate descent, with closed-form iterations

$$u_i = \frac{H_{ii}v_i}{\sigma^2 + \sum_j H_{ji}^2 v_j^2}$$

$$w_i = \frac{1}{1 - u_i H_{ii} v_i}$$

$$v_i = \frac{\alpha_i u_i H_{ii} w_i}{\mu + \sum_j \alpha_j H_{ij}^2 u_j^2 w_j}$$

- May not always converge to the global optimum
- Computationally expensive with high time complexity
- Cannot incorporate node state info
- Must be rerun for each instance of \mathbf{H}

Connectionist Approach

- ▶ Use neural networks to learn the optimal power allocation $\mathbf{p}(\mathbf{H}, \mathbf{X})$
- ▶ GNNs are good candidates to model this allocation
 - ⇒ CSI \mathbf{H} as a **weighted adjacency** matrix of a **directed graph**
 - ⇒ NSI \mathbf{X} as a signal supported at the **nodes**

Connectionist Approach

- ▶ Use neural networks to learn the optimal power allocation $\mathbf{p}(\mathbf{H}, \mathbf{X})$
- ▶ GNNs are good candidates to model this allocation
 - ⇒ CSI \mathbf{H} as a **weighted adjacency** matrix of a **directed graph**
 - ⇒ NSI \mathbf{X} as a **signal supported at the nodes**
- ▶ $\mathbf{p}(\mathbf{H}, \mathbf{X}) = \Psi(\mathbf{H}, \mathbf{X}; \Theta)$, where Ψ is a K -layered **GNN**
 - ⇒ Θ is the set of **trainable weights**

Connectionist Approach

- ▶ Use neural networks to learn the optimal power allocation $\mathbf{p}(\mathbf{H}, \mathbf{X})$
- ▶ GNNs are good candidates to model this allocation
 - ⇒ CSI \mathbf{H} as a weighted adjacency matrix of a directed graph
 - ⇒ NSI \mathbf{X} as a signal supported at the nodes
- ▶ $\mathbf{p}(\mathbf{H}, \mathbf{X}) = \Psi(\mathbf{H}, \mathbf{X}; \Theta)$, where Ψ is a K -layered GNN
 - ⇒ Θ is the set of trainable weights
- ▶ Supervised Training: Learn by using WMMSE output as training signals
- ▶ Unsupervised Training: Learn using Sum-rate as the optimization objective

REGNN

- ▶ Standard layered GNN architecture

$$\mathbf{z}_l = \text{ReLU} \left(\sum_{f=0}^{F_l} \gamma_{lf} \mathbf{H}^f \mathbf{z}_{l-1} \right) \quad \mathbf{z}_0 = \mathbf{X}, \quad \Phi(\mathbf{H}, \mathbf{X}; \boldsymbol{\gamma}) = \mathbf{z}_L$$

REGNN

- ▶ Standard layered GNN architecture

$$\mathbf{z}_l = \text{ReLU} \left(\sum_{f=0}^{F_l} \gamma_{lf} \mathbf{H}^f \mathbf{z}_{l-1} \right) \quad \mathbf{z}_0 = \mathbf{X}, \quad \Phi(\mathbf{H}, \mathbf{X}; \boldsymbol{\gamma}) = \mathbf{z}_L$$

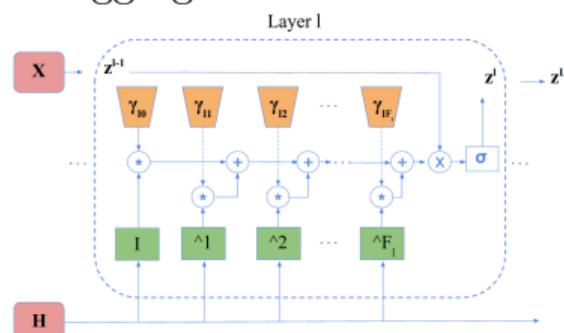
- ▶ Graph filter $\sum_{f=0}^{F_l} \gamma_{lf} \mathbf{H}^f$ combines data within F_l -hop neighborhoods
- ▶ Alternate local linear aggregation of data with pointwise non-linearity
- ▶ Learn the best weights in the local aggregation of data

REGNN

- ▶ Standard layered GNN architecture

$$\mathbf{z}_l = \text{ReLU} \left(\sum_{f=0}^{F_l} \gamma_{lf} \mathbf{H}^f \mathbf{z}_{l-1} \right) \quad \mathbf{z}_0 = \mathbf{X}, \quad \Phi(\mathbf{H}, \mathbf{X}; \boldsymbol{\gamma}) = \mathbf{z}_L$$

- ▶ Graph filter $\sum_{f=0}^{F_l} \gamma_{lf} \mathbf{H}^f$ combines data within F_l -hop neighborhoods
- ▶ Alternate local linear aggregation of data with pointwise non-linearity
- ▶ Learn the best weights in the local aggregation of data



Eisen-Ribeiro TSP'20

IGCNet

- ▶ Compute pairwise influence (interference) of each neighbor

$$\gamma_{ji}^k = MLP1(H_{ji}, H_{ij}, x_j, H_{jj}, \beta_j^{k-1}) \quad \forall i, j \in \mathbb{N}_i$$

IGCNet

- ▶ Compute pairwise influence (interference) of each neighbor

$$\gamma_{ji}^k = MLP1(H_{ji}, H_{ij}, x_j, H_{jj}, \beta_j^{k-1}) \quad \forall i, j \in \mathbb{N}_i$$

- ▶ Local non-linear aggregation of neighborhood interference

$$\alpha_i^k = CONCAT(MAX_j(\gamma_{ji}), \sum_j \gamma_{ji}) \quad \forall i, j \in \mathbb{N}_i$$

IGCNet

- ▶ Compute pairwise influence (interference) of each neighbor

$$\gamma_{ji}^k = \text{MLP1}(H_{ji}, H_{ij}, x_j, H_{jj}, \beta_j^{k-1}) \quad \forall i, j \in \mathbb{N}_i$$

- ▶ Local non-linear aggregation of neighborhood interference

$$\alpha_i^k = \text{CONCAT}(\text{MAX}_j(\gamma_{ji}), \sum_j \gamma_{ji}) \quad \forall i, j \in \mathbb{N}_i$$

- ▶ Learn policy based on combination of channel with interference

$$\beta_i^k = \text{MLP2}(\alpha_i^k, H_{ii}, \beta_i^{k-1}, x_i) \quad \forall i$$

IGCNet

- ▶ Compute pairwise influence (interference) of each neighbor

$$\gamma_{ji}^k = \text{MLP1}(H_{ji}, H_{ij}, x_j, H_{jj}, \beta_j^{k-1}) \quad \forall i, j \in \mathbb{N}_i$$

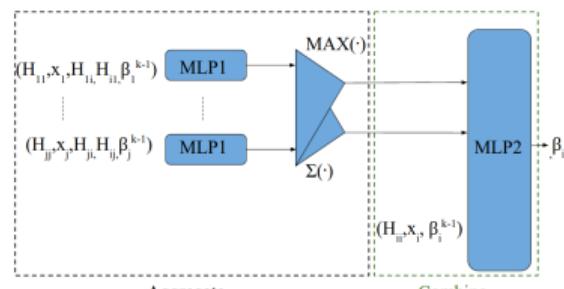
- ▶ Local non-linear aggregation of neighborhood interference

$$\alpha_i^k = \text{CONCAT}(\text{MAX}_j(\gamma_{ji}), \sum_j \gamma_{ji}) \quad \forall i, j \in \mathbb{N}_i$$

- ▶ Learn policy based on combination of channel with interference

$$\beta_i^k = \text{MLP2}(\alpha_i^k, H_{ii}, \beta_i^{k-1}, x_i) \quad \forall i$$

Layer k - Node i



Shen *et al.*, Globecom'19

Graph-based *Unfolding* Scheme

► Iterative algorithm

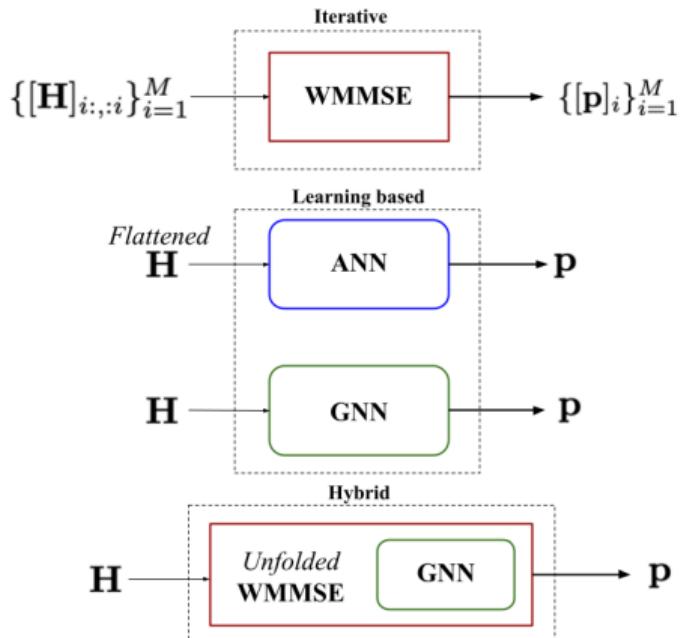
- ⇒ Near-optimal
- ⇒ Time-consuming
- ⇒ Greedy

► Learnable models

- ⇒ MLP ignores graph structure
- ⇒ GNN ignores domain info.

► Hybrid model

- ⇒ Iterations as layers
- ⇒ Embedded graph model
- ⇒ *Inherits greediness*



Algorithm Unrolling

- ▶ Iterative algorithms are long cascades of iterative steps
 - ⇒ Good performance but slow and/or expensive
- ▶ Each step computes variables of interest from a set of parameters

Algorithm Unrolling

- ▶ Iterative algorithms are long cascades of iterative steps
⇒ Good performance but slow and/or expensive
- ▶ Each step computes variables of interest from a set of parameters
- ▶ Limit number of iterations ⇒ suboptimal performance
- ▶ Need more efficient parameter update for faster convergence

Algorithm Unrolling

- ▶ Iterative algorithms are long cascades of iterative steps
⇒ Good performance but slow and/or expensive
- ▶ Each step computes variables of interest from a set of parameters
- ▶ Limit number of iterations ⇒ suboptimal performance
- ▶ Need more efficient parameter update for faster convergence
- ▶ Algorithm Unrolling - learn from data Monga, Li & Eldar, 2019 arxiv, 2021 IEEE SPM)
⇒ Supervised/Un-supervised gradient feedback
- ▶ Iterations ⇒ layers, Parameters ⇒ neural networks

Algorithm Unrolling

- ▶ Iterative algorithms are long cascades of iterative steps
⇒ Good performance but slow and/or expensive
- ▶ Each step computes variables of interest from a set of parameters
- ▶ Limit number of iterations ⇒ suboptimal performance
- ▶ Need more efficient parameter update for faster convergence
- ▶ Algorithm Unrolling - learn from data
Monga, Li & Eldar, 2019 arxiv, 2021 IEEE SPM)
⇒ Supervised/Un-supervised gradient feedback
- ▶ Iterations ⇒ layers, Parameters ⇒ neural networks
- ▶ GNNs have built-in scalability and permutation invariance
- ▶ More interpretable operations, easy to follow update trajectory
- ▶ Easy to fall back to classical solutions
- ▶ Once trained, can be used off-the-shelf ⇒ Effective for online solutions

Hybrid Method - UWMMSE

- ▶ UWMMSE update rules at layer k

$$\textcolor{blue}{a}^{(k)} = \Psi(\mathbf{H}; \boldsymbol{\theta}_a^{(k)}), \quad \textcolor{blue}{b}^{(k)} = \Psi(\mathbf{H}; \boldsymbol{\theta}_b^{(k)}) \quad (1)$$

$$u_i^{(k)} = \frac{h_{ii}v_i^{(k-1)}}{\sigma^2 + \sum_j h_{ij}^2 v_j^{(k-1)}}, \quad (2)$$

$$w_i^{(k)} = \frac{\textcolor{blue}{a}_i^{(k)}}{1 - u_i^{(k)} h_{ii} v_i^{(k-1)}} + \textcolor{blue}{b}_i^{(k)}, \quad (3)$$

$$v_i^{(k)} = \alpha \left(\frac{u_i^{(k)} h_{ii} w_i^{(k)}}{\sum_j h_{ji}^2 u_j^{(k)} u_j^{(k)} w_j^{(k)}} \right), \quad (4)$$

- ▶ **v** and **u** as TX and RX variables
- ▶ **w** as a tunable parameter
- ▶ $\textcolor{blue}{a} = 1, \textcolor{blue}{b} = 0$ yields classical solution

Hybrid Method - UWMMSE

- UWMMSE update rules at layer k

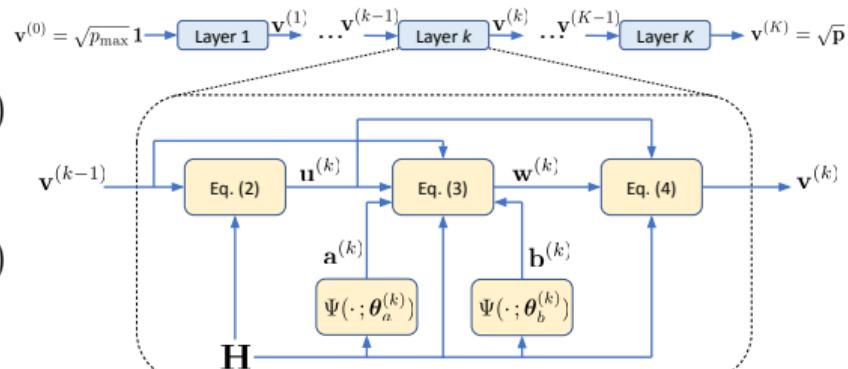
$$\mathbf{a}^{(k)} = \Psi(\mathbf{H}; \boldsymbol{\theta}_a^{(k)}), \quad \mathbf{b}^{(k)} = \Psi(\mathbf{H}; \boldsymbol{\theta}_b^{(k)}) \quad (1) \quad \blacktriangleright k^{th} \text{ layer of the model:}$$

$$u_i^{(k)} = \frac{h_{ii}v_i^{(k-1)}}{\sigma^2 + \sum_j h_{ij}^2 v_j^{(k-1)}}, \quad (2)$$

$$w_i^{(k)} = \frac{\mathbf{a}_i^{(k)}}{1 - u_i^{(k)} h_{ii} v_i^{(k-1)}} + \mathbf{b}_i^{(k)}, \quad (3)$$

$$v_i^{(k)} = \alpha \left(\frac{u_i^{(k)} h_{ii} w_i^{(k)}}{\sum_j h_{ji}^2 u_j^{(k)} u_j^{(k)} w_j^{(k)}} \right), \quad (4)$$

- \mathbf{v} and \mathbf{u} as TX and RX variables
- \mathbf{w} as a tunable parameter
- $\mathbf{a} = 1, \mathbf{b} = 0$ yields classical solution

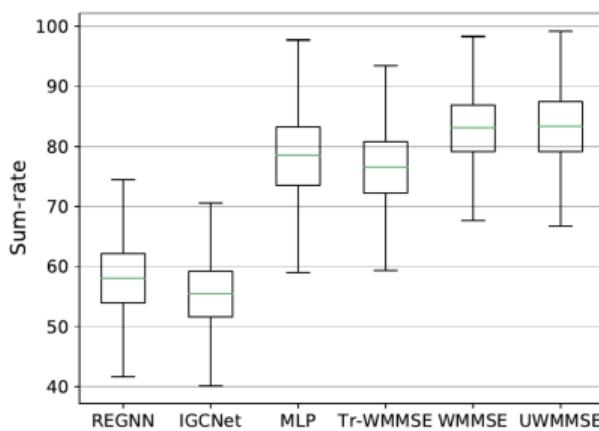


Simulation Results - 1

- ▶ Random geometric graph with M node pairs
- ▶ Path loss and Rayleigh fading

▶ Performance Comparison

- ⇒ Network size $M = 20$;
- ⇒ $K = 4, K_{\max} = 100$



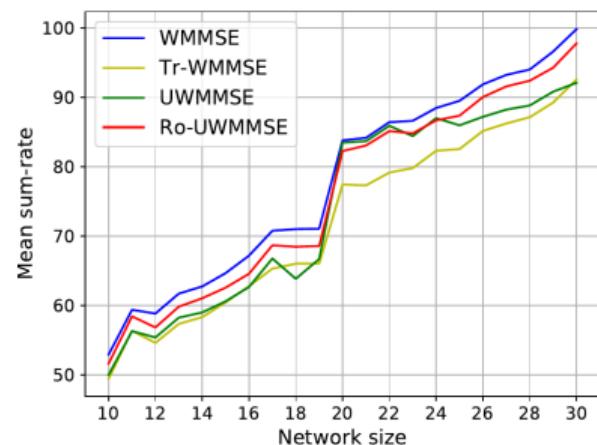
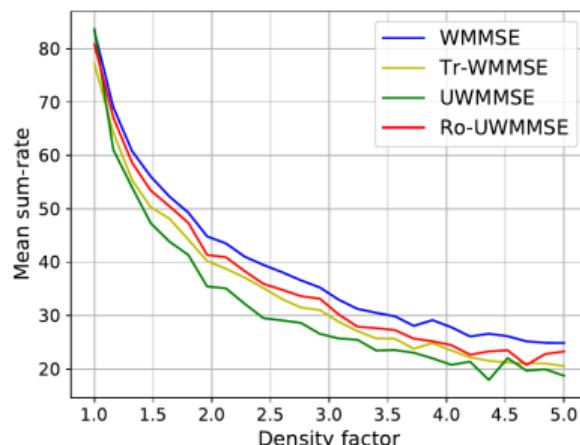
▶ Time Comparison

Algorithm	Training time (m)	Test sum-rate	Test time (ms)
WMMSE	-	82.94	16
Tr-WMMSE	-	76.49	1.0
MLP	0.5	78.17	3.2
REGNN	15	57.92	2.5
IGCNet	5	55.30	3
UWMMSE	15	83.21	2.0

WMMSE: Shi *et al.*, TSP'11, MLP: Sun *et al.*, TSP'18, REGNN: Eisen-Ribeiro TSP'20, IGCNet: Shen *et al.*, Globecom'21,
UWMMSE: Chowdhury *et al.*, ICASSP'21, TWC'21

Simulation Results - 2

- ▶ Simulating dynamic network topologies
 - ⇒ Nodes in motion
 - ⇒ Insertion / Deletion of nodes
- ▶ Variation in Spatial Density
- ▶ Variation in Network Size



Chowdhury *et al.*, ICASSP'21, ICASSP'22, TWC'21

Optimal Power Allocation & Beamforming MIMO Case

Arindam Chowdhury *et al.*, IEEE MILCOM'21, Asilomar'23, IEEE Trans.
Wireless Communications 2023
Arindam Chowdhury, PhD thesis, Rice Univ. 2024

System Model

- ▶ Ad hoc network with M transmitter-receiver pairs (nodes)
- ▶ Transmitters have T antennas, receivers have R antennas
- ▶ Transmitter i has an associated receiver $r(i) \forall i \in \{1, M\}$

System Model

- ▶ Ad hoc network with M transmitter-receiver pairs (nodes)
- ▶ Transmitters have T antennas, receivers have R antennas
- ▶ Transmitter i has an associated receiver $r(i) \forall i \in \{1, M\}$
- ▶ Channel State Information (CSI) tensor $\mathcal{H} \in \mathbb{R}^{M \times M \times R \times T}$
 - ⇒ Encodes channel characteristics
 - ⇒ $[\mathcal{H}]_{ji::} = \mathbf{H}_{ji} \in \mathbb{R}^{R \times T}$ represents a MIMO channel from i to $r(j)$
 - ⇒ Channel between Tx-antenna k and Rx-antenna l is given by

$$[\mathbf{H}_{ji}]_{lk} = H_{jilk} = H_{jilk}^P H_{jilk}^F(t)$$

⇒ where $H_{jilk}^P \propto \text{dist}(i, r(j))^{-k}$ for all l, k and $H_{jilk}^F \sim \text{Rayleigh}(\alpha)$

System Model

- ▶ Ad hoc network with M transmitter-receiver pairs (nodes)
- ▶ Transmitters have T antennas, receivers have R antennas
- ▶ Transmitter i has an associated receiver $r(i) \forall i \in \{1, M\}$
- ▶ Channel State Information (CSI) tensor $\mathcal{H} \in \mathbb{R}^{M \times M \times R \times T}$
 - ⇒ Encodes channel characteristics
 - ⇒ $[\mathcal{H}]_{ji::} = \mathbf{H}_{ji} \in \mathbb{R}^{R \times T}$ represents a MIMO channel from i to $r(j)$
 - ⇒ Channel between Tx-antenna k and Rx-antenna l is given by

$$[\mathbf{H}_{ji}]_{lk} = H_{jilk} = H_{jilk}^P H_{jilk}^F(t)$$

- ⇒ where $H_{jilk}^P \propto \text{dist}(i, r(j))^{-k}$ for all l, k and $H_{jilk}^F \sim \text{Rayleigh}(\alpha)$
- ▶ Transmitter beamformer tensor $\mathcal{V} \in \mathbb{R}^{M \times T \times d}$
 - ⇒ $[\mathcal{V}]_i = \mathbf{V}_i \in \mathbb{R}^{T \times d}$ transmits signal $\mathbf{s}_i \in \mathbb{R}^d$ at node i

Problem Description

Given the CSI tensor \mathcal{H} , and a network utility function $u(\mathcal{H}, \mathcal{V}, \mathbf{p})$, determine the optimal power allocation \mathbf{p} and \mathcal{V} , under power constraint $p_i \leq P_{\max}$

Problem Description

Given the CSI tensor \mathcal{H} , and a network utility function $u(\mathcal{H}, \mathcal{V}, \mathbf{p})$, determine the optimal power allocation \mathbf{p} and \mathcal{V} , under power constraint $p_i \leq P_{\max}$

- ▶ Network utility: **sum rate** across nodes $\sum_{i=1}^M \alpha_i c_i$
- ▶ **Data rate** at receiver i is given by (for noise variance σ^2)

$$c_i(\mathcal{H}, \mathcal{V}) = \log_2 \det \left(\mathbf{I} + \mathbf{H}_{ii} \mathbf{V}_i \mathbf{V}_i^\top \mathbf{H}_{ii}^\top (\sigma^2 \mathbf{I} + \sum_{j \neq i} \mathbf{H}_{ij} \mathbf{V}_j \mathbf{V}_j^\top \mathbf{H}_{ij}^\top)^{-1} \right)$$

where $\text{Tr} \left(\mathbf{V}_i \mathbf{V}_i^\top \right) \leq p_i$

Problem Description

Given the CSI tensor \mathcal{H} , and a network utility function $u(\mathcal{H}, \mathcal{V}, \mathbf{p})$, determine the optimal power allocation \mathbf{p} and \mathcal{V} , under power constraint $p_i \leq P_{\max}$

- ▶ Network utility: **sum rate** across nodes $\sum_{i=1}^M \alpha_i c_i$
- ▶ **Data rate** at receiver i is given by (for noise variance σ^2)

$$c_i(\mathcal{H}, \mathcal{V}) = \log_2 \det \left(\mathbf{I} + \mathbf{H}_{ii} \mathbf{V}_i \mathbf{V}_i^\top \mathbf{H}_{ii}^\top (\sigma^2 \mathbf{I} + \sum_{j \neq i} \mathbf{H}_{ij} \mathbf{V}_j \mathbf{V}_j^\top \mathbf{H}_{ij}^\top)^{-1} \right)$$

$$\text{where } \text{Tr} \left(\mathbf{V}_i \mathbf{V}_i^\top \right) \leq p_i$$

- ▶ Classical solution is WMMSE
- ▶ Seeking a function $\Psi(\mathcal{H})$ by combining **unrolling** with **GNN**

Hybrid Method - UWMMSE

- UWMMSE update rules at layer k (\mathbf{U} is $R \times d$, \mathbf{W} $d \times d$, \mathbf{V} $T \times d$)

$$\mathbf{a}^{(k)} = \Psi(\bar{\mathcal{H}}; \theta_a), \quad \mathbf{b}^{(k)} = \Psi(\bar{\mathcal{H}}; \theta_b), \quad (1)$$

$$\mathbf{U}_i^{(k)} = \left(\sum_{j \neq i} \mathbf{H}_{ij} \mathbf{V}_j^{(k-1)} \mathbf{V}_j^{(k-1)\top} \mathbf{H}_{ij}^\top + \sigma^2 \mathbf{I}_R \right)^{-1} \mathbf{H}_{ii} \mathbf{V}_i^{(k-1)} \quad \forall i \quad (2)$$

$$\mathbf{W}_i^{(k)} = [\mathbf{a}^{(k)}]_i (\mathbf{I}_d - \mathbf{U}_i^{(k)\top} \mathbf{H}_{ii} \mathbf{V}_j^{(k-1)})^{-1} + [\mathbf{b}^{(k)}]_i \quad \forall i \quad (3)$$

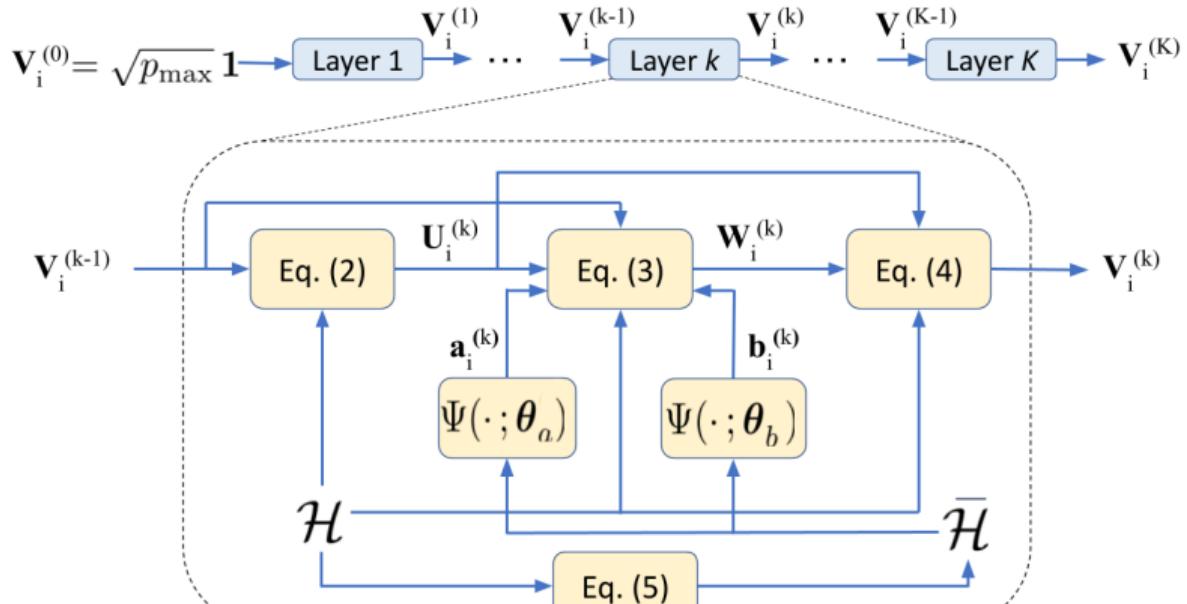
$$\mathbf{V}_i^{(k)} = \beta \left(\left(\sum_{j \neq i} \mathbf{H}_{ij}^\top \mathbf{U}_j^{(k)} \mathbf{W}_j^{(k)} \mathbf{U}_j^{(k)\top} \mathbf{H}_{ij} + \epsilon \mathbf{I}_T \right)^{-1} \mathbf{H}_{ii}^\top \mathbf{U}_i^{(k)} \mathbf{W}_i^{(k)} \right) \quad \forall i \quad (4)$$

- β is a clipper to enforce power constraint

$$\beta(\mathbf{X}) = \begin{cases} \mathbf{X}, & \text{if } \text{Tr}(\mathbf{X}\mathbf{X}^\top) \leq P_{\max}, \\ \mathbf{X} \cdot \frac{\sqrt{P_{\max}}}{\|\mathbf{X}\|_F}, & \text{otherwise,} \end{cases}$$

Block Diagram

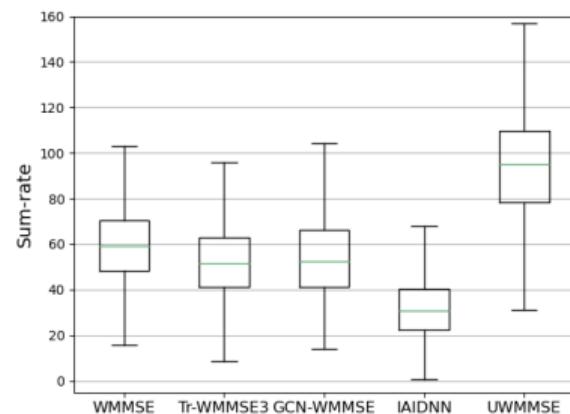
- k^{th} layer of the model:



- Complexity: $12h + RT + 1$ trainable weights
- Easy to train, likely to generalize

Simulation Results - 1

- ▶ Random geometric graph with M node pairs
- ▶ Path loss & fading: Rayleigh, Rician, Network size $M = 20$
- ▶ **Performance Comparison**
- ▶ **Time Comparison**



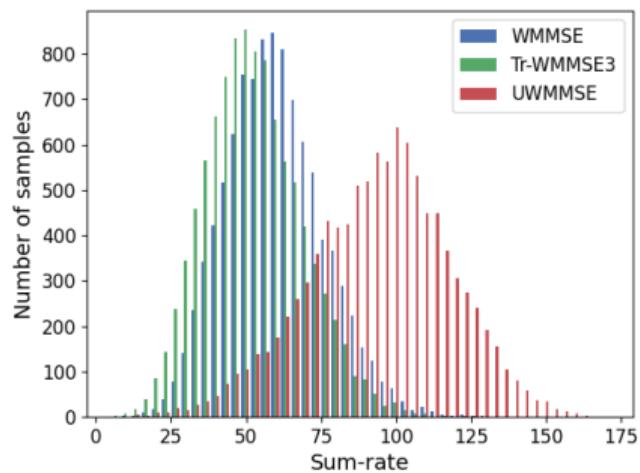
Algorithm	Training time (min)	Test time (sec)
WMMSE	-	1.305
Tr-WMMSE	-	0.047
IAIDNN	~ 10	0.64
GCN-WMMSE	~ 21	1.365
UWMMSE	~ 35	0.054

WMMSE: Shi *et al.*, TSP'11, IAIDNN: Hu *et al.* TWC'21, GCN-WMMSE: Schynol-Pesavento, JSAC'23,

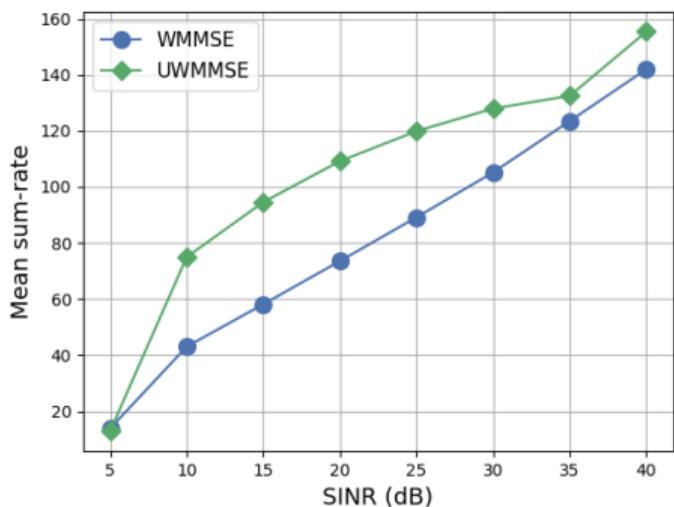
UWMMSE: Chowdhury *et al.*, MILCOM'21, Asilomar'23, TWC'23; Chowdhury, PhD thesis, Rice, 2024

Simulation Results -2

► Performance Comparison



- Generalization performance
⇒ Over SINR



Chowdhury *et al.*, MILCOM'21, TWC'23, Asilomar'23, PhD thesis (Rice)'24

Optimal Power Allocation - Federated Learning

Boning Li *et al.* IEEE ICASSP 2022, IEEE Trans. Wireless Communications 2023;

Boning Li, PhD thesis - Rice Univ., 2024

Code at: <https://github.com/bl166/WirelessFL-PDG>

Power allocation for wireless FL

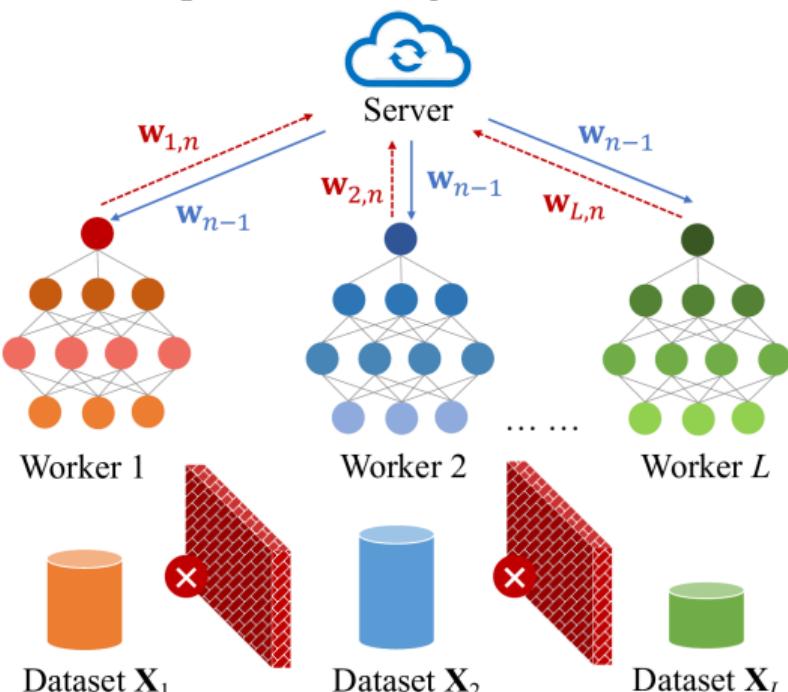


Critical step: upload local updates

How much **transmit power** should local workers use?

Power allocation for wireless FL

Compared to the just-discussed SISO & MISO power allocation cases ...

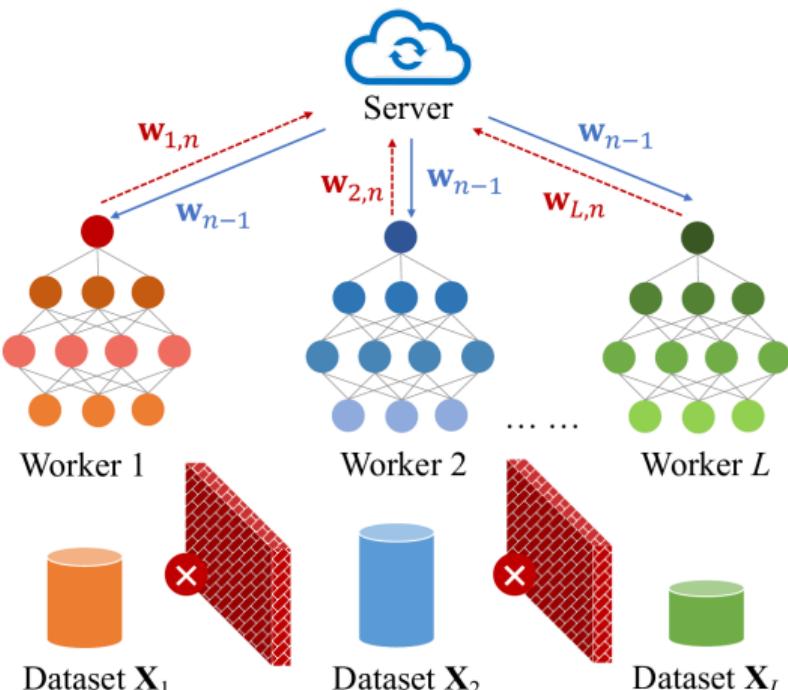


FL case is more challenging:

- ▶ Additional non-convex constraints on FL-specific requirements, e.g., delay and energy
- ▶ Ultimate goal of improving FL performance is indirect to the communication objective

Power allocation for wireless FL

Compared to the just-discussed SISO & MISO power allocation cases ...



FL case is more challenging:

- ▶ Additional non-convex constraints on FL-specific requirements, e.g., delay and energy
⇒ Primal-dual (PD) algorithm enhanced by graph learning
- ▶ Ultimate goal of improving FL performance is indirect to the communication objective
⇒ Local data heterogeneity

Problem formulation

Determine the power allocation policy $p^* : \mathbb{R}^{L \times L} \rightarrow \mathbb{R}^L$ that solves the following optimization problem¹, subject to bounds on **transmission rate**, **energy efficiency**, and **power**

$$\begin{aligned} p^* &= \underset{p}{\operatorname{argmax}} \ g(\mathbb{E}_{\mathbf{H} \sim \mathcal{H}} [\text{PSR}(\mathbf{p}, \mathbf{H})]), \\ \text{s.t. } &\textcolor{blue}{r_{0,i}} \leq \mathbb{E}_{\mathbf{H} \sim \mathcal{H}} [R_i(\mathbf{p}, \mathbf{H}) \mid \textcolor{red}{p_i > 0}], \\ &\textcolor{green}{e_{0,i}} \leq \mathbb{E}_{\mathbf{H} \sim \mathcal{H}} \left[\frac{R_i(\mathbf{p}, \mathbf{H})}{p_i + P_{c,i}} \mid \textcolor{red}{p_i > 0} \right], \forall i, \\ &\mathbf{p} = p(\mathbf{H}) \in [0, P_{\max}], \quad \forall \mathbf{H}, \end{aligned}$$

¹PSR: Packet success rate, $\text{PSR} = \exp(-m/\text{SINR})$

Problem formulation

Parameterize the power policy with learnable parameters Θ , st $p_\psi(\mathbf{H}) = \Psi(\mathbf{H}; \Theta)$, and restate P1 ...

$$p^* = \underset{p}{\operatorname{argmax}} \ g(\mathbb{E}_{\mathbf{H} \sim \mathcal{H}} [\text{PSR}(\mathbf{p}, \mathbf{H})])$$

$$\begin{aligned} \text{s.t. } r_{0,i} &\leq \mathbb{E}_{\mathbf{H} \sim \mathcal{H}} [R_i(\mathbf{p}, \mathbf{H}) \mid p_i > 0], \\ e_{0,i} &\leq \mathbb{E}_{\mathbf{H} \sim \mathcal{H}} \left[\frac{R_i(\mathbf{p}, \mathbf{H})}{p_i + P_{c,i}} \mid p_i > 0 \right], \forall i, \\ \mathbf{p} &= p(\mathbf{H}) \in [0, P_{\max}], \quad \forall \mathbf{H}, \end{aligned}$$

...in a manner that is amenable to a **Primal-Dual (PD) solution:**

$$P_\psi^* = \max_{\Theta, \mathbf{y}, \mathbf{r}, \mathbf{e}} \ g(\mathbf{y}), \quad (\text{P2})$$

$$\text{s.t. } \mathbf{y} \leq \mathbb{E}[\text{PSR}(\mathbf{p}_\psi, \mathbf{H})],$$

$$r_i \leq \mathbb{E}[R_i(\mathbf{p}_\psi, \mathbf{H}) \mid p_{\psi,i} > 0],$$

$$e_i \leq \mathbb{E} \left[\frac{R_i(\mathbf{p}_\psi, \mathbf{H})}{p_{\psi,i} + P_c} \mid p_{\psi,i} > 0 \right],$$

$$r_i \in [r_{0,i}, +\infty),$$

$$e_i \in [e_{0,i}, +\infty), \quad \forall i,$$

$$\mathbf{p}_\psi = p_\psi(\mathbf{H}) \in [0, P_{\max}], \quad \forall \mathbf{H}.$$

PD learning

- ▶ (P1) has a **zero duality gap**.
- ▶ (P2)'s duality gap depends on the **expressiveness of Ψ** .

The Lagrangian of (P2)

$$\mathcal{L}_\psi(\boldsymbol{\Theta}, \mathbf{y}, \mathbf{r}, \mathbf{e}, \boldsymbol{\lambda}_y, \boldsymbol{\lambda}_r, \boldsymbol{\lambda}_e) = g(\mathbf{y}) + \boldsymbol{\lambda}_y^\top (\mathbb{E}[f_y] - \mathbf{y}) + \boldsymbol{\lambda}_r^\top (\mathbb{E}_c[f_r] - \mathbf{r}) + \boldsymbol{\lambda}_e^\top (\mathbb{E}_c[f_e] - \mathbf{e})$$

motivates iterative gradient updates to:

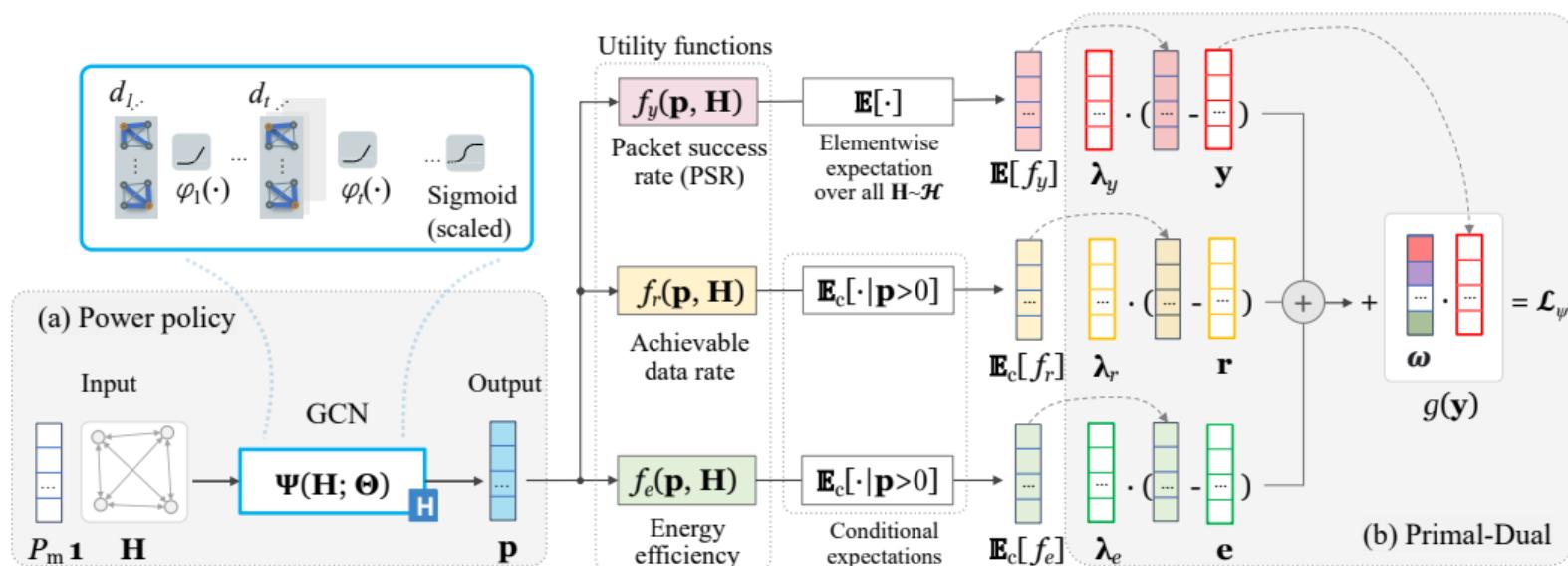
1. Learnable parameters $\boldsymbol{\Theta}$;
2. Primal variables \mathbf{y} , \mathbf{r} , and \mathbf{e} ;
3. Dual variables $\boldsymbol{\lambda}_y$, $\boldsymbol{\lambda}_r$, and $\boldsymbol{\lambda}_e$.

Choosing GCN as Ψ constitutes our primal-dual graph convolutional (PDG) power network.

GCN Based Power Policy - Learning

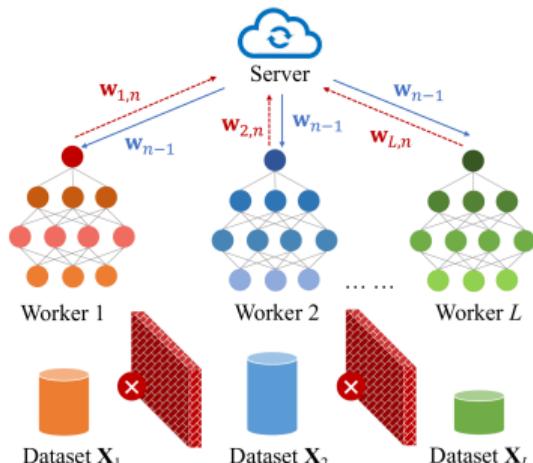
Two-stage solution with **two separate learning models**.

1. Before FL, train a **power allocation policy model**.
2. During FL, apply the policy model to upload **local FL models** in each FL iteration, which then update the **global FL model**.



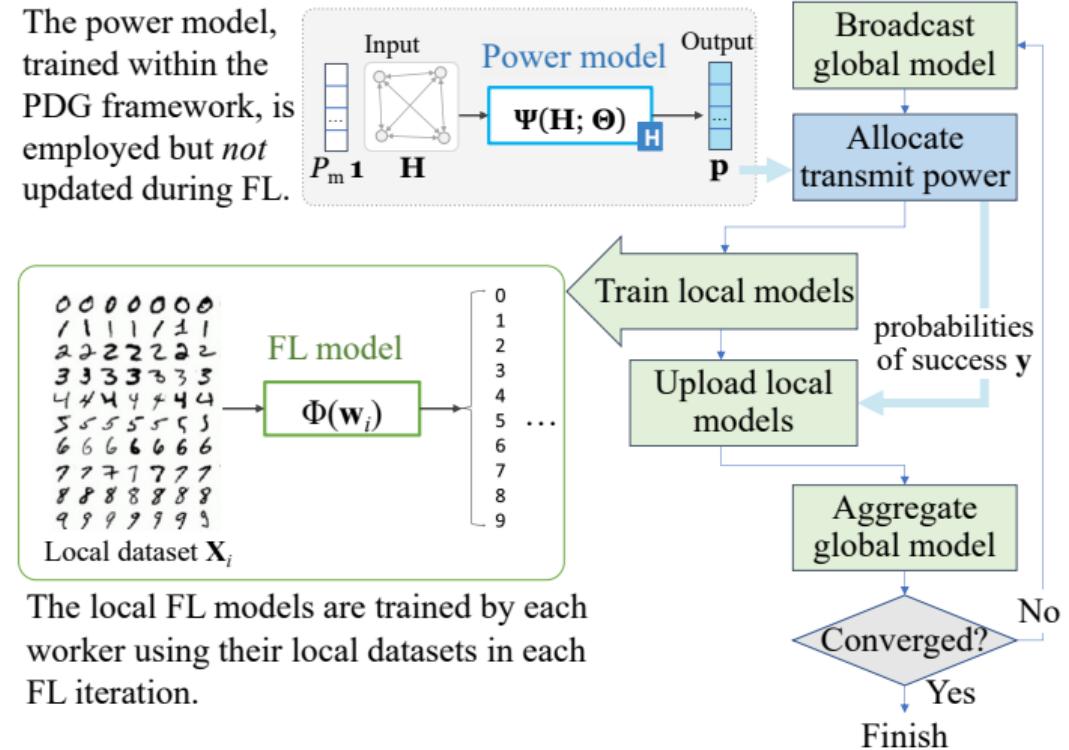
FL Pipeline with GCN-based power policy

FL system:



FL pipeline with power allocation policy:

The power model, trained within the PDG framework, is employed but *not* updated during FL.

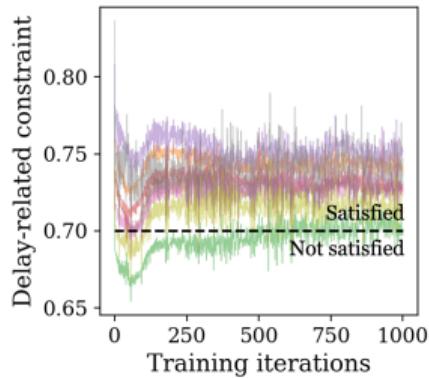


The local FL models are trained by each worker using their local datasets in each FL iteration.

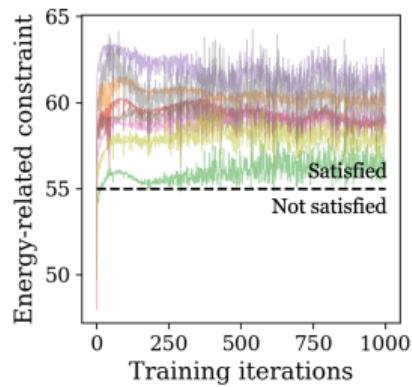
PD learning curves for the power policy

Learning curves² of PDG demonstrate convergence to

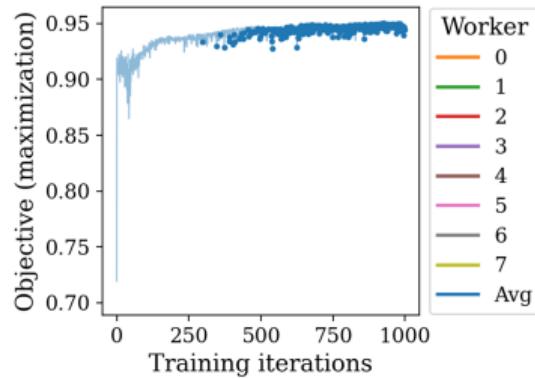
- (a) delay constraint, (b) energy constraint, and (c) objective PSR.



(a)



(b)



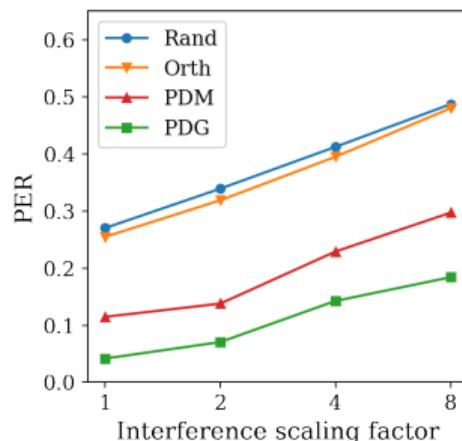
(c)

²Constraint constants r_0 and e_0 are annotated as dashed lines in (a) and (b). Larger markers in (c) are where all workers satisfy both constraints.

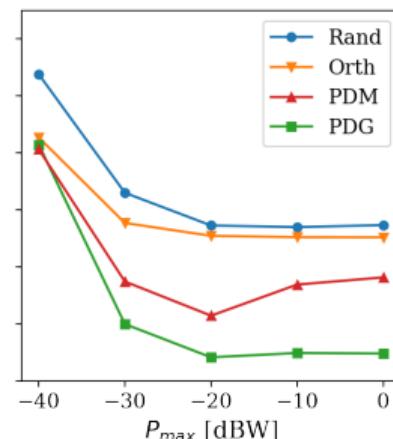
Communication proxy

Performance comparison (**system-level transmission error rate**) of PDG to other power allocation methods under different network configs:

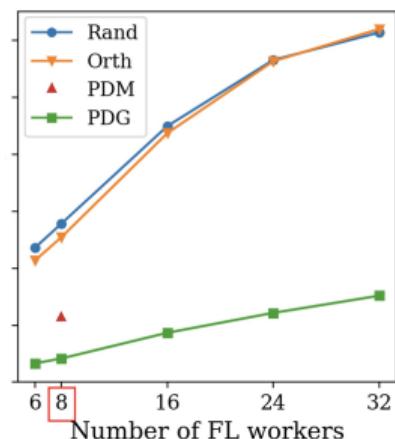
(a) Interference strength



(b) Max power value



(c) Network size



PDG ensures **more accurate transmissions** than the topology- agnostic learning-based PDM and other rule-based power methods.

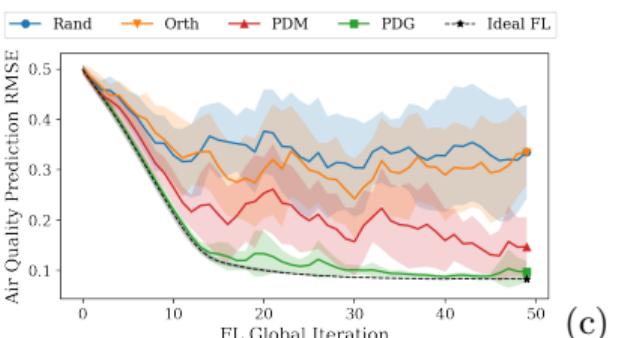
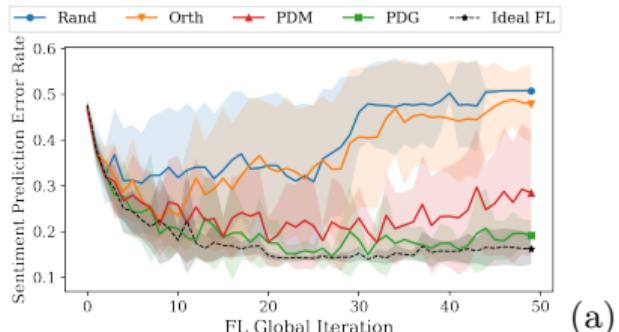
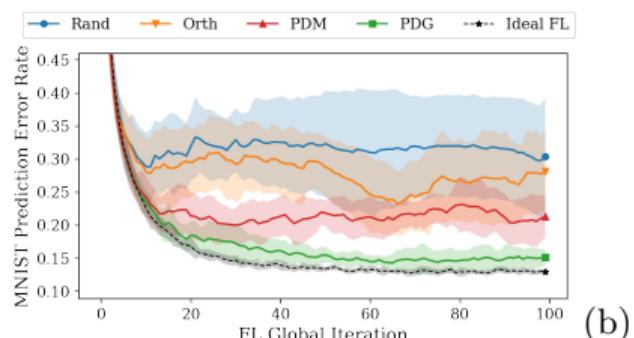
Orth: Chen *et al.*, TWC'20

FL performance on I.I.D. data

Tests on FL benchmark tasks: (a) NLP: IMDb sentiment classification, (b) MNIST digit classification, (c) regression: Air quality prediction.

Figures show **global FL** validation errors vs completed FL iterations.

PDG consistently yields the **best FL performance**, close to ideal.



Boning Li *et al.* ICASSP'22, TWC'23

Other Applications at the PHY Layer

Other PHY Layer Problems

Soft Interference Cancellation

- ▶ Learn conditional distro via DNN
Deep-SIC, SICNet
Schelzinger+ TWC'21, TVT'
- ▶ ViterbiNet: Decoding w/o CSI
Schelzinger+ TWC'20

Medium Access

- ▶ RL / DQN based: S. Wang+,
TCCN'18, A. Kumar+ ICASSP'21,
M.Zhang+ Comp. Comm.'22

Learning bounds: Habi+ TSP'23

Code design: Bourtsoulatze+ TCCN'19,
Makkuva+ ICML'21, Chahine+ TWC'24

Other PHY Layer Problems

Soft Interference Cancellation

- ▶ Learn conditional distro via DNN
Deep-SIC, SICNet
Schelzinger+ TWC'21, TVT'
- ▶ ViterbiNet: Decoding w/o CSI
Schelzinger+ TWC'20

Medium Access

- ▶ RL / DQN based: S. Wang+,
TCCN'18, A. Kumar+ ICASSP'21,
M.Zhang+ Comp. Comm.'22

Learning bounds: Habi+ TSP'23

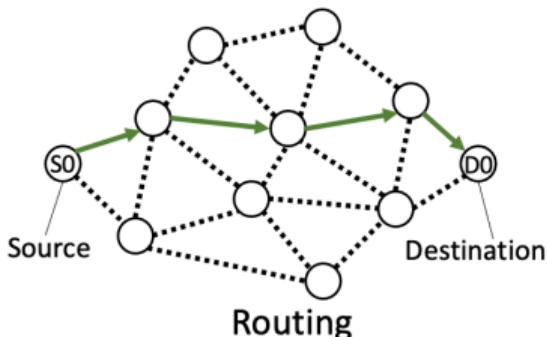
Code design: Bourtsoulatze+ TCCN'19,
Makkuva+ ICML'21, Chahine+ TWC'24

Signal / Modulation Classification

- ▶ DeepMusic - Elbir IEEE Sens.
Letts.'20
- ▶ RF Signal Classification: Yi Shi *et al.*, IEEE DySpan'19
- ▶ Modulation Classification:
 - ⇒ W. Zhang *et al.*, INFOCOM'21
 - ⇒ Zhang *et al.*, MILCOM'24
 - ⇒ Svensson+, MILCOM'24
 - ⇒ Snoap *et al.*, MILCOM'23
 - ⇒ ...

Part III: Graph-based ML for Wireless Networking

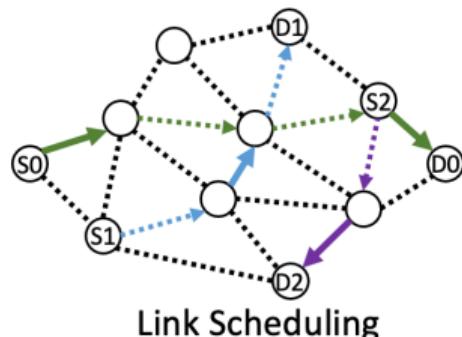
Fundamental wireless networking tasks



Source

Destination

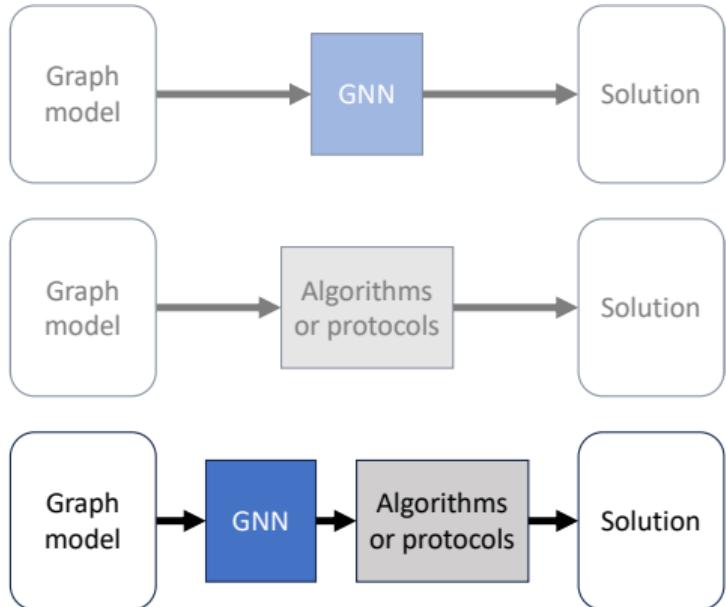
Routing



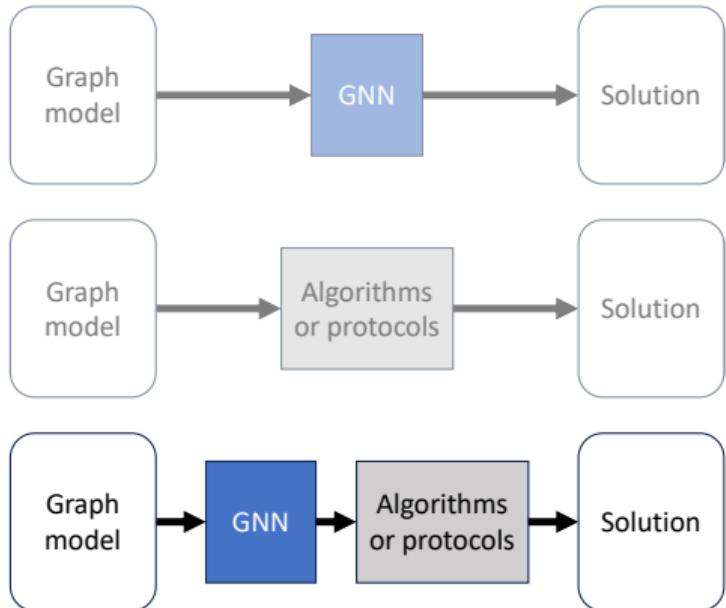
Link Scheduling

- ▶ **Routing:** send packets from source(s) to destination(s) through relay nodes
 - ⇒ Path finding: 1-to-1 (unicast), 1-to-many: multi-cast, broadcast
 - ⇒ Orchestration: cluster head election, virtual backbone establishment
- ▶ **Link scheduling:** decide which links to be activated in each time slot
 - ⇒ MaxWeight scheduling, carrier sensing multiple access (CSMA)
- ▶ **Combinatorial & discrete nature**
- ▶ **Distributed solutions** are preferred (our focus)

High-level Ideas



High-level Ideas



Optimization problems with hard constraints

- ▶ Max-Weight link scheduling ^a
- ▶ Context-aware routing^b & offloading^c
- ▶ Repetitive combinatorial optimization^d

^a Z. Zhao, G. Verma, C. Rao, A. Swami and S. Segarra, "Link Scheduling Using Graph Neural Networks," in IEEE Trans. on Wireless Comms., vol. 22, no. 6, pp. 3997-4012, June 2023

^b Z. Zhao, B. Radojičić, G. Verma, A. Swami, S. Segarra, Biased Backpressure Routing Using Link Features and Graph Neural Networks, IEEE Trans. on Machine Learning In Comms. and Netw., 2024.

^c Z. Zhao, J. Perazzone, G. Verma and S. Segarra, "Congestion-Aware Distributed Task Offloading in Wireless Multi-Hop Networks Using Graph Neural Networks," IEEE ICASSP, 2024, pp. 8951-8955.

^d Z. Zhao, A. Swami, S. Segarra, "Graph-based Deterministic Policy Gradient for Repetitive Combinatorial Optimization Problems," ICLR 2023

Combinatorial Optimization Problem (COP)

Typical formulation

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \quad \text{COP}$$

s.t. *Discrete constraint on nodes or edges*
*Constraints defined on Graph,
 Hypergraph, or Simplicial Complex*

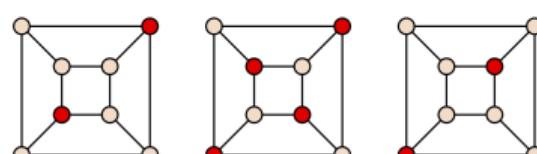
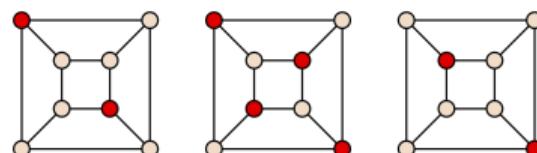
Characters

- ▶ Input: a graph with cost vector \mathbf{c}
- ▶ Decision variables \mathbf{x}
 - ⇒ Discrete (integer) constraints
 - ⇒ Relational constraints
- ▶ Minimize total cost
- ▶ Non-convex, often **NP-hard!**

Maximum Weighted Independent Set

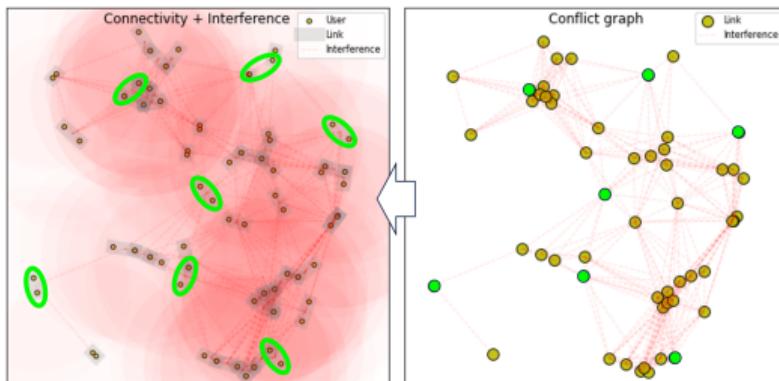
$$\mathbf{v}^* = \operatorname{argmax}_{\mathbf{v} \subseteq \{0,1\}^{|\mathcal{V}|}} \mathbf{u}^\top \mathbf{v} \quad (5a)$$

$$\text{s.t. } \mathbf{v}_i + \mathbf{v}_j \leq 1, \forall (i, j) \in \mathcal{E}. \quad (5b)$$



Source: Wikipedia – Maximal independent set

Link scheduling: MWIS formulation



- ▶ Conflict graph
 - ⇒ Vertex: a wireless link
 - ⇒ Edge: conflict relationship between two links
 - ⇒ Vertex weight: utility of scheduling a link
- ▶ Optimal schedule: non-conflicting links with maximum total utility

Repetitive Combinatorial Optimization Problem (R-COP)



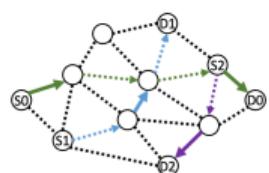
- ▶ Graph-based Markov decision process
 - ⇒ Network state as a weighted graph $(\mathcal{V}(t), \mathcal{E}(t), \mathbf{c}(t))$
 - ⇒ Network state of $t + 1$ depends on decisions $\mathbf{x}(t)$
 - ⇒ Decision $\mathbf{x}(t)$ found by solving a COP on $(\mathcal{V}(t), \mathcal{E}(t), \mathbf{c}(t))$
 - ⇒ Cost vector $\mathbf{c}(t)$ changes rapidly compared to topology $(\mathcal{V}(t), \mathcal{E}(t))$

Repetitive Combinatorial Optimization Problem (R-COP)



- ▶ Graph-based Markov decision process
 - ⇒ Network state as a weighted graph $(\mathcal{V}(t), \mathcal{E}(t), \mathbf{c}(t))$
 - ⇒ Network state of $t + 1$ depends on decisions $\mathbf{x}(t)$
 - ⇒ Decision $\mathbf{x}(t)$ found by solving a COP on $(\mathcal{V}(t), \mathcal{E}(t), \mathbf{c}(t))$
 - ⇒ Cost vector $\mathbf{c}(t)$ changes rapidly compared to topology $(\mathcal{V}(t), \mathcal{E}(t))$
- ▶ Many applications

Routing & Scheduling in
communication networks



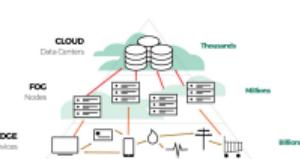
Multi-object tracking in
computer vision



Vehicle routing problems in
distribution networks

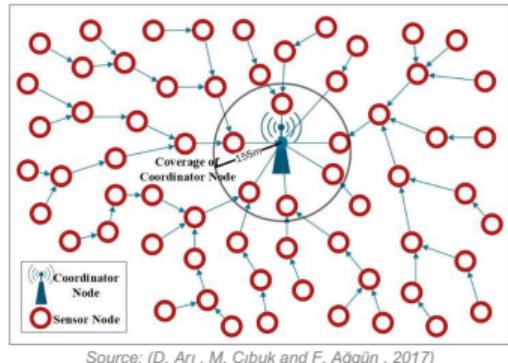
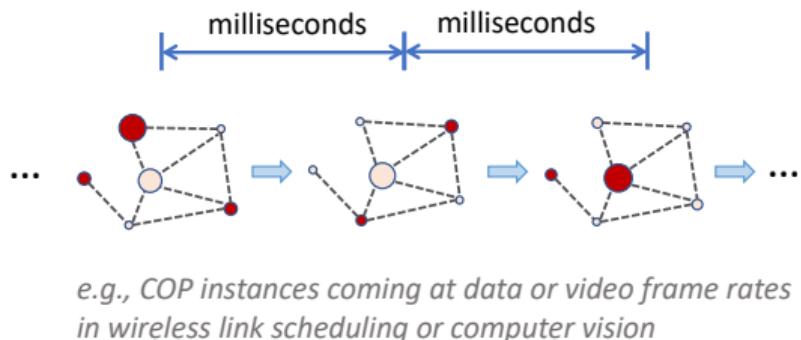


Resource allocation & job scheduling in
cloud, fog, edge computing



- ▶ Practical restrictions: **limited runtime** and/or **distributed execution**

Restrictions on runtime and distributed execution



- ▶ Centralized COP solver
 - ⇒ High communication overhead to collect full network state to a server
 - ⇒ High computational complexity, scale up quickly by network size
 - ⇒ Single point of failure
- ▶ Distributed COP solver for **scalability** and **robustness**
 - ⇒ Fast & robust execution using only neighborhood information (exchange)

Why not just let a GNN directly output solution?

Graph neural networks (GNNs)

- ▶ Distributed & fast execution
- ▶ Generalize to different topologies
- ▶ Unable to encode relational constraints in COPs, e.g., $\mathbf{v}_i + \mathbf{v}_j \leq 1 , \forall (v_i, v_j) \in \mathcal{E}$.

Why not just let a GNN directly output solution?

Graph neural networks (GNNs)

- ▶ Distributed & fast execution
- ▶ Generalize to different topologies
- ▶ Unable to encode relational constraints in COPs, e.g., $\mathbf{v}_i + \mathbf{v}_j \leq 1 , \forall (v_i, v_j) \in \mathcal{E}$.

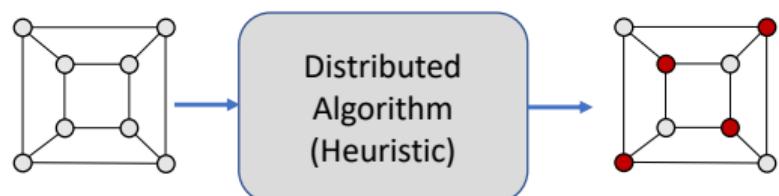


Identical input → identical prediction

Constraint violations

Conventional distributed heuristics

- ▶ Distributed & fast execution
- ▶ Generalize to different topologies
- ▶ Guaranteed feasibility
- ▶ Large optimality gap



Tie-breaking rules



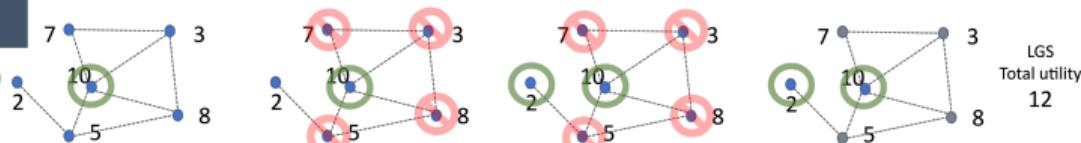
Example: MWIS problem on a regular graph, where every node has identical weight

Link scheduling: distributed local greedy solver

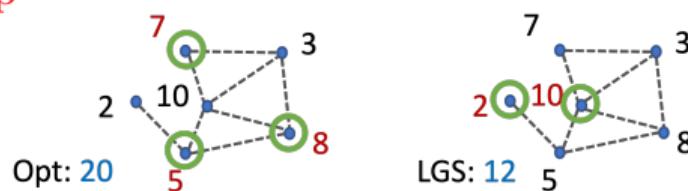
- Local Greedy Solver (LGS)³ operate on the conflict graph, $\mathcal{O}(\log |\mathcal{V}|)$
 - ⇒ A vertex (link) is scheduled if its utility > all neighbors
 - ⇒ A vertex (link) is muted if one of its neighbors is scheduled
 - ⇒ Otherwise, it enters the next iteration until all nodes are decided

An illustrative example

Local greedy solver (LGS)

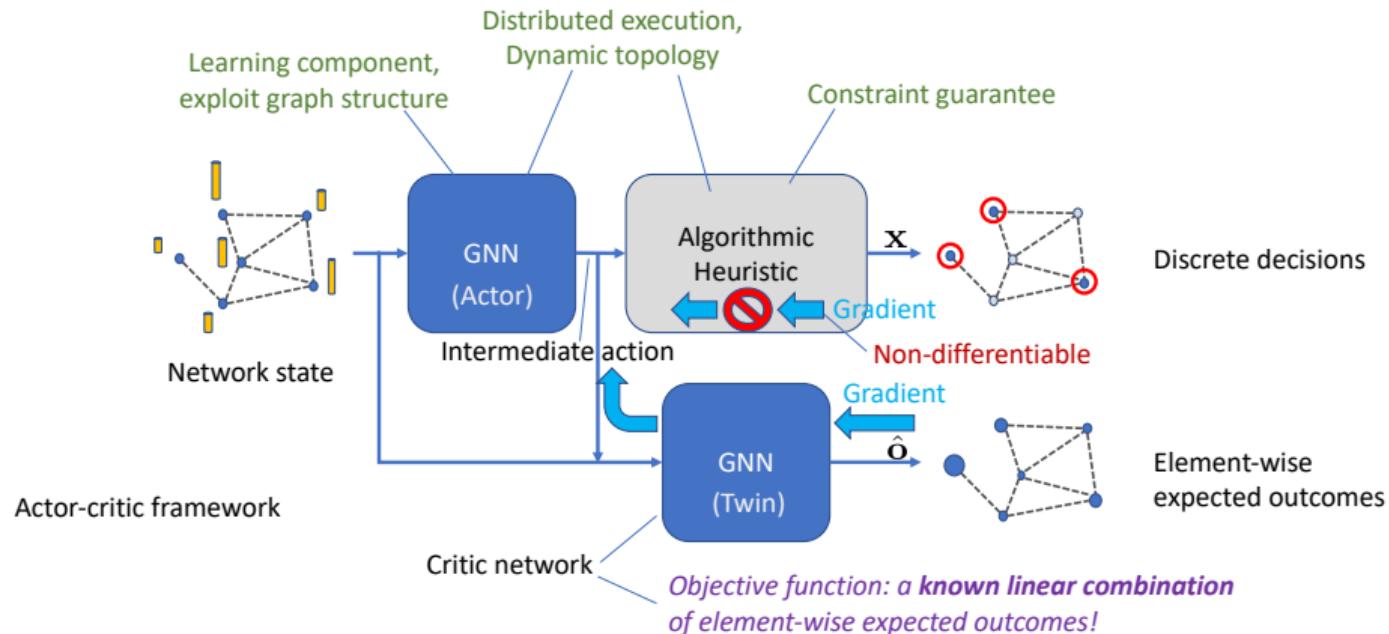


- Large optimality gap



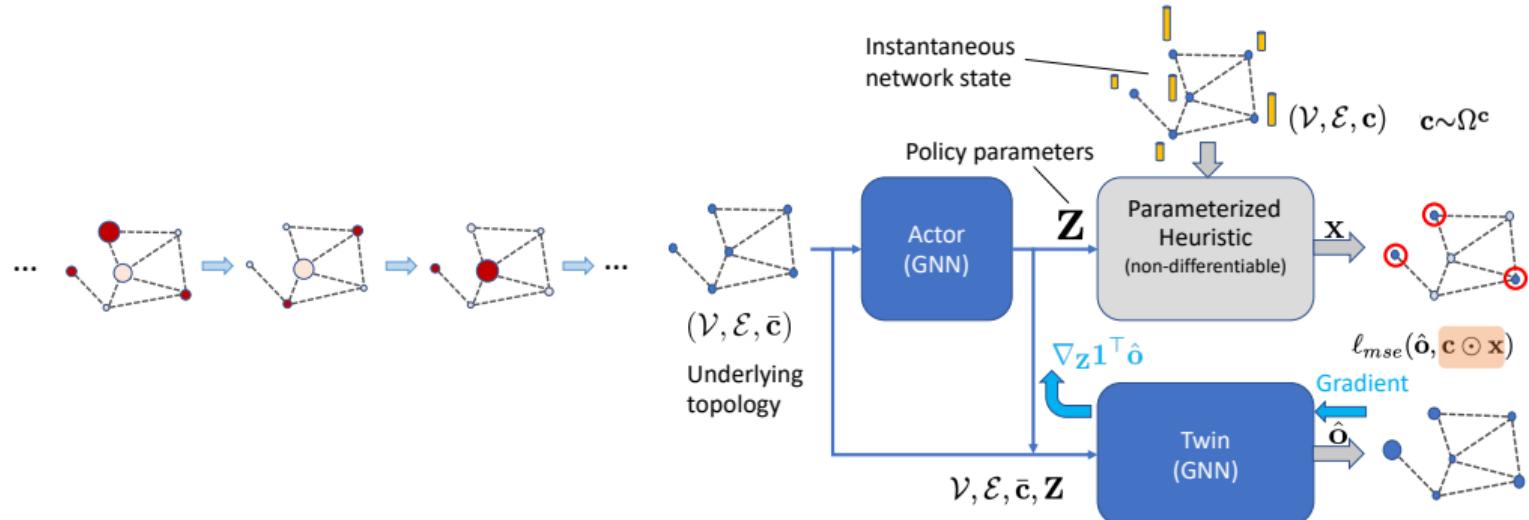
³C. Joo and N. B. Shroff, “Local greedy approximation for scheduling in multihop wireless networks,” IEEE Trans. on Mobile Computing, vol. 11, no. 3, pp. 414–426, 2012

GDPG-Twin: a general actor-critic framework for R-COP



- ▶ Actor GNN exploits graph structure
- ▶ Algorithmic heuristic guarantee correctness (relational constraints)
- ▶ Twin GNN bridges the non-differentiability gap of algorithmic heuristic

Independent R-COP



- ▶ Goal: reduce **optimality gap** with minimal overhead
 - ⇒ Optimize each COP instance individually, **ignore inter-state dependency**
- ▶ GNN encodes the underlying topology, embeddings reused for many time steps
- ▶ Expected element-wise outcome $\hat{\mathbf{o}} \approx \mathbf{o} = \mathbb{E}(\mathbf{c} \odot \mathbf{x})$
- ▶ Gradient on intermediate action $\nabla_{\mathbf{Z}} \mathbf{1}^\top \hat{\mathbf{o}} \approx \nabla_{\mathbf{Z}} \mathbb{E}(\mathbf{c}^\top \mathbf{x})$

Independent R-MWIS

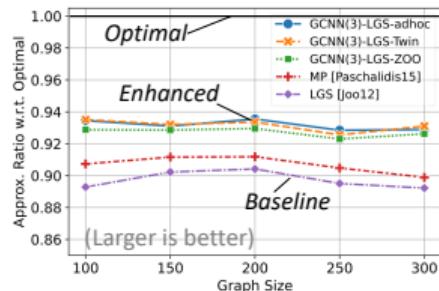


Figure 1: Approximation ratios (Larger is better) of the vanilla and GCNN-enhanced distributed heuristics for MWIS problem (max), w.r.t. the optimal solver.

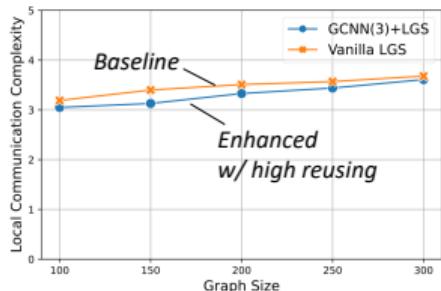


Figure 3: Average local communication complexity of GCNN-enhanced and vanilla LGS-MWIS solvers per instance, in rounds, excluding the GCNN ($N = \infty$).

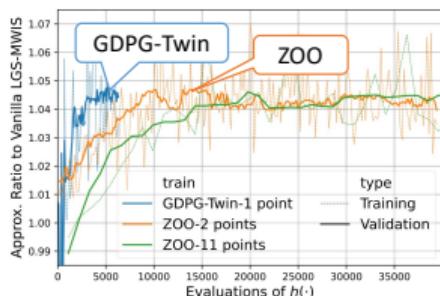


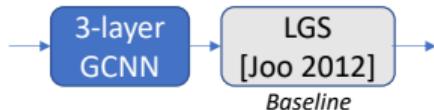
Figure 8: Performance trajectories of GCNN-enhanced LGS-MWIS trained by GPG-Twin and ZOOs with 2-point and 11-point gradient estimations. Larger is better. GPG-Twin needs fewer evaluations of $h(\cdot)$.

Approximation ratio

Execution local complexity

Training complexity

Benchmark: ZOO (zeroth-order optimization)



- ▶ Tested on 500 random graphs from Erdős–Rényi model
- ▶ Baseline: LGS⁴, Benchmark: Zeroth-order optimization (ZOO)

⁴C. Joo and N. B. Shroff, "Local Greedy Approximation for Scheduling in Multihop Wireless Networks," in IEEE Trans. on Mobile Computing, vol. 11, no. 3, pp. 414-426, March 2012.

Generalize to more Independent R-COPs

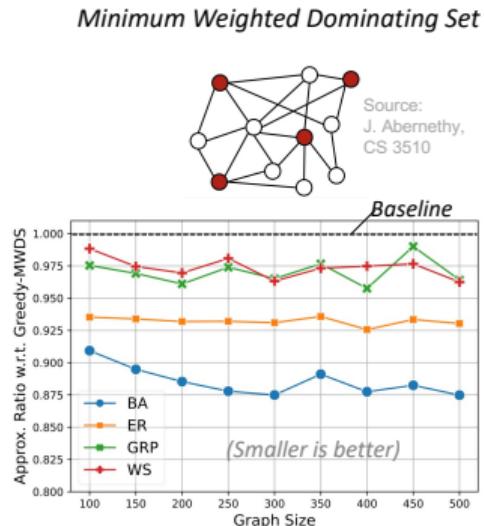
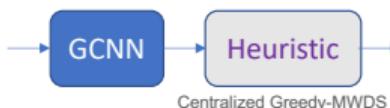


Figure 2: Approximation ratio (Smaller is better) of the GCNN-enhanced w.r.t. the vanilla Greedy-MWDS for MWDS problem (min) on 4 sets of random graphs.



Node Weighted Steiner Tree

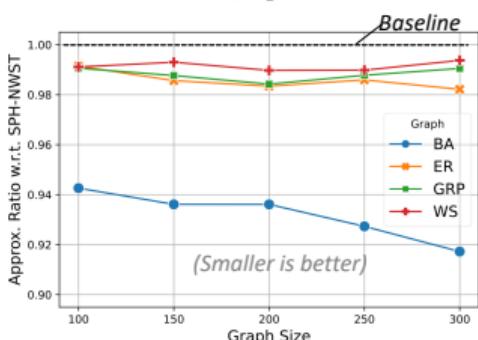
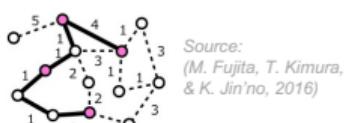
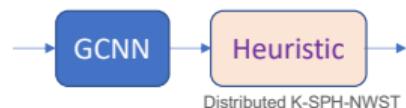


Figure 5: Approximation ratio (Smaller is better) of the GCNN-enhanced w.r.t. vanilla K-SPH-NWST for NWST problem on 4 sets of random graphs. NWST is a minimization (min) problem.



Minimum Weighted Connected Dominating Set

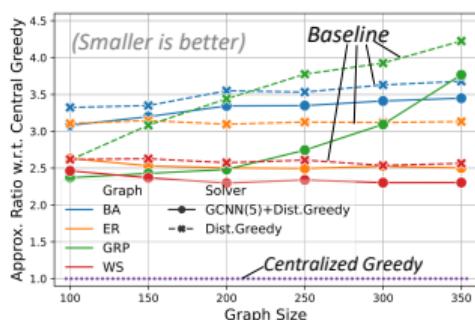
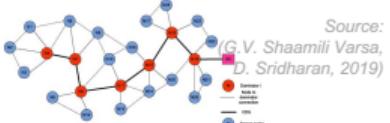
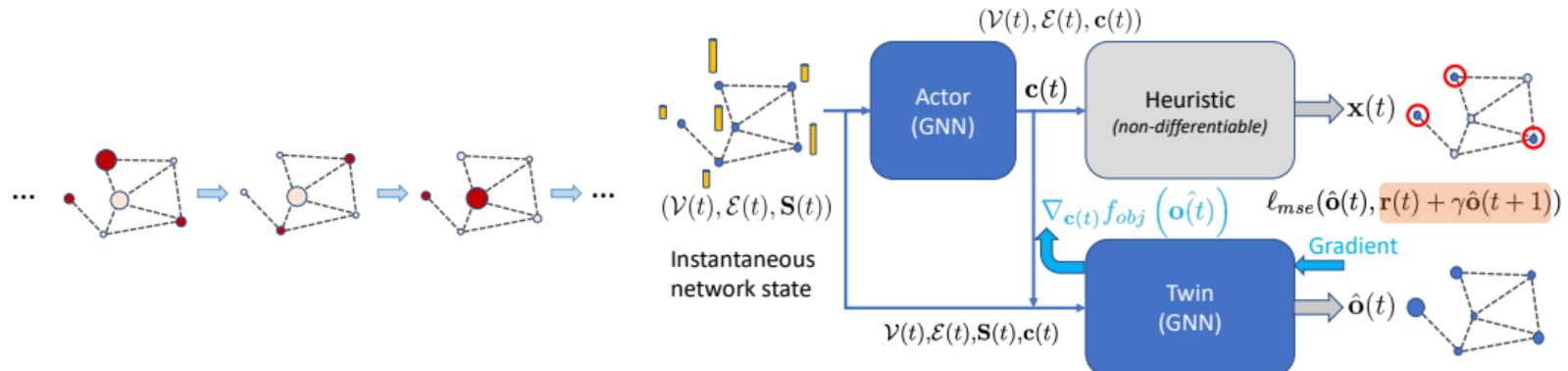


Figure 6: Approximation ratios (Smaller is better) of the vanilla and GCNN-enhanced distributed heuristics w.r.t. a centralized heuristic for MWCDS problem on 4 sets of random graphs. MWCDS is a min. problem.

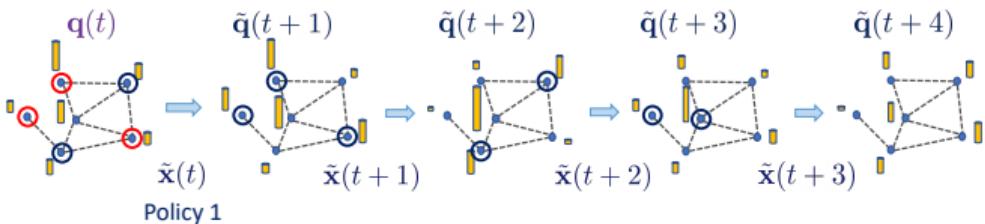


R-COP in graph-based Markov decision process



- ▶ Goal: optimize **long-term** system-level objective
⇒ Inter-state dependency MUST be considered
- ▶ GNN encodes network state $(\mathcal{V}(t), \mathcal{E}(t), \mathbf{S}(t))$ into cost vector $\mathbf{c}(t)$ in each time step
⇒ Consider future element-wise rewards
- ▶ Expected element-wise outcome $\hat{\mathbf{o}}(t) \approx \mathbf{o}(t) = \mathbb{E} [\mathbf{r}(t) + \gamma \hat{\mathbf{o}}(t+1)]$
- ▶ Gradient on intermediate action $\nabla_{\mathbf{c}(t)} f_{obj} (\hat{\mathbf{o}}(t))$, f_{obj} is a linear function

Delay-oriented link scheduling



$$\omega^* = \operatorname{argmax}_{\omega} \mathbb{E}_{i \in \mathcal{V}, t \leq T} [\mathbf{q}_i(t)] \quad (6a)$$

$$\text{s.t. } \mathbf{q}(t+1) = \mathbf{q}(t) + \mathbf{a}(t) - \mathbf{x}(t) \odot \min(\mathbf{l}(t), \mathbf{q}(t)), \quad (6b)$$

$$\mathbf{x}(t) = h(\mathcal{V}, \mathcal{E}, \mathbf{q}(t), \mathbf{l}(t), \mathbf{a}(t); \omega). \quad (6c)$$

$\mathbf{l}(t)$ link rates, $\mathbf{q}(t)$ queue lengths, $\mathbf{a}(t)$ new packet arrivals

- ▶ The ML pipeline is supposed to improve delay on centralized networks
- ▶ GDPG-Twin can do the same job as *ad-hoc RL scheme*^a at $\frac{1}{5}$ computational cost

^aZ. Zhao, G. Verma, A. Swami and S. Segarra, "Delay-Oriented Distributed Scheduling Using Graph Neural Networks," IEEE ICASSP 2022, pp. 8902-8906

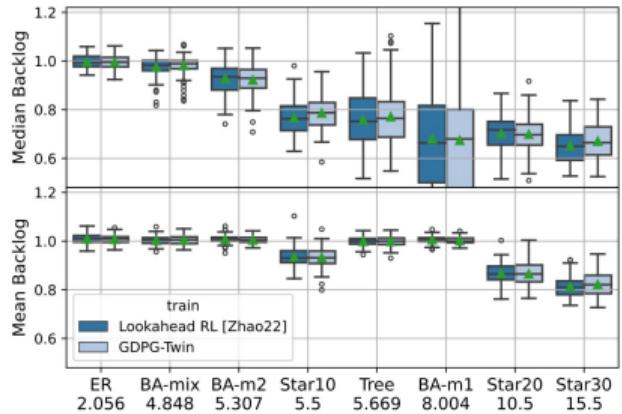
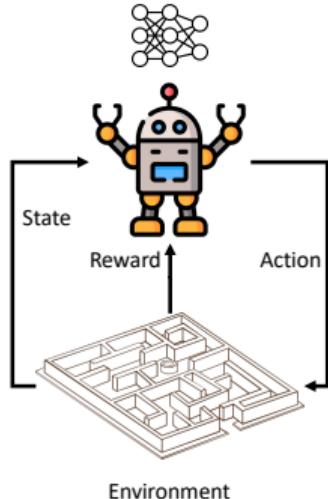


Figure 7: GDPG-Twin achieves similar network-wide mean and medium backlogs (smaller is better) of lookahead RL (Zhao et al., 2022b) in training a distributed link scheduler, using only $\frac{1}{5}$ evaluations of $h(\cdot)$ of it.

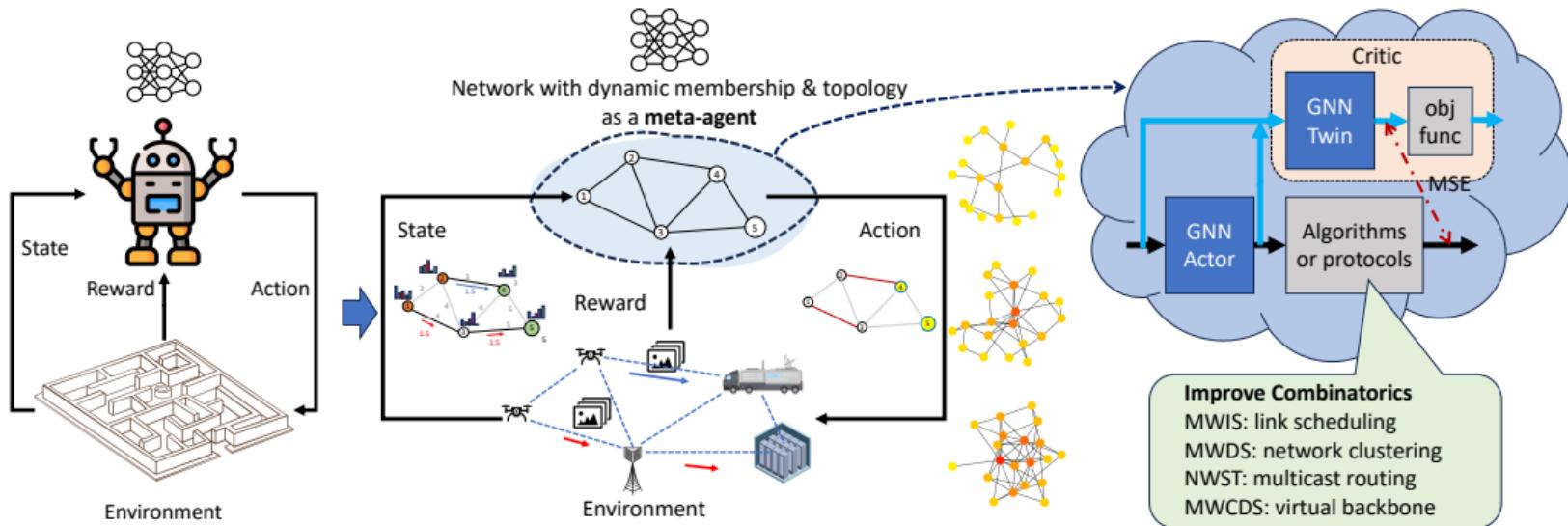
Recap on GPG-Twin for R-COPs

- ▶ Single-agent reinforcement learning for: scalar action & reward, state in regular domain

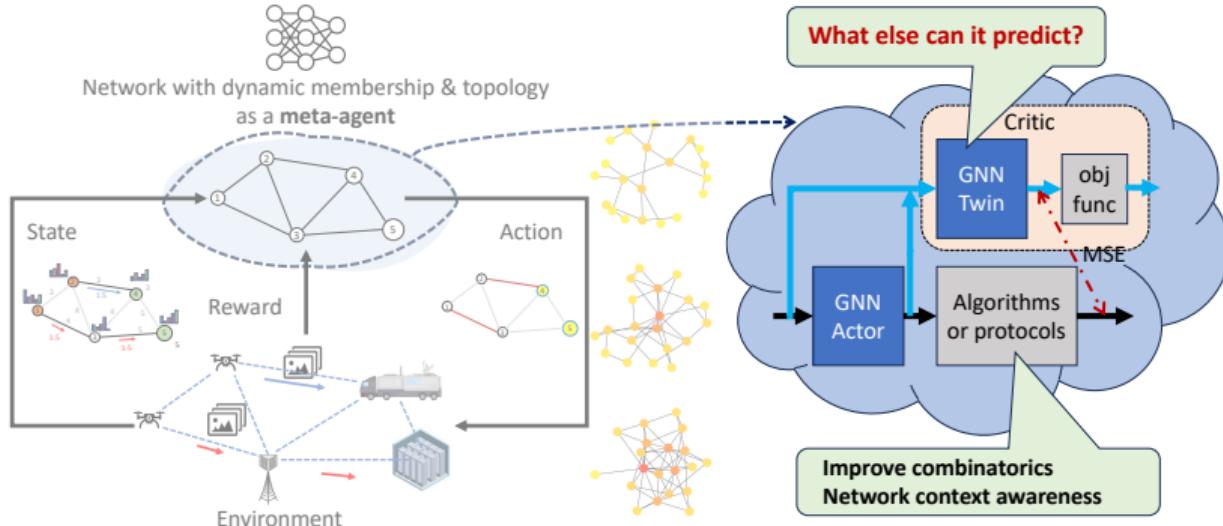


Recap on GDPG-Twin for R-COPs

- ▶ Single-agent reinforcement learning for: **scalar** action & reward, state in **regular** domain



- ▶ GPG-Twin as a general reinforcement learning framework for distributed networks
 - ⇒ High-dimensional **parallel** action, reward, & state in **irregular** (graph) domain
 - ⇒ Generalize to dynamic graphs thanks to shared core model in GNN
 - ⇒ Follow engineered rules, leveraging domain knowledge



► Network context awareness

- ⇒ Backpressure routing in wireless ad-hoc networks^a
- ⇒ Congestion-aware distributed task offloading^b

^aZ. Zhao, B. Radojičić, G. Verma, A. Swami, S. Segarra, Biased Backpressure Routing Using Link Features and Graph Neural Networks, IEEE Trans. on Machine Learning In Comms. and Netw., 2024.

^bZ. Zhao, J. Perazzone, G. Verma and S. Segarra, "Congestion-Aware Distributed Task Offloading in Wireless Multi-Hop Networks Using Graph Neural Networks," IEEE ICASSP, 2024, pp. 8951-8955.

Fast & Differentiable network simulators

Network Digital Twin for Fast KPI prediction⁵⁶

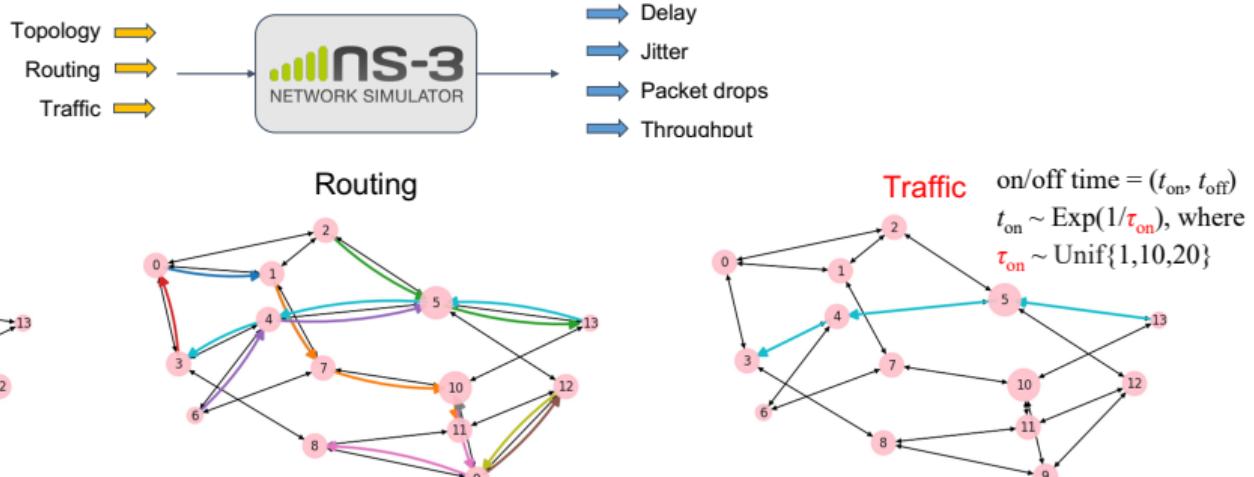


Credit: Boning Li

⁵B. Li, T. Efimov, A. Kumar, J. Cortes, G. Verma, A. Swami, and S. Segarra. "Learnable Digital Twin for Efficient Wireless Network Evaluation." In IEEE MILCOM, pp. 661-666., 2023.

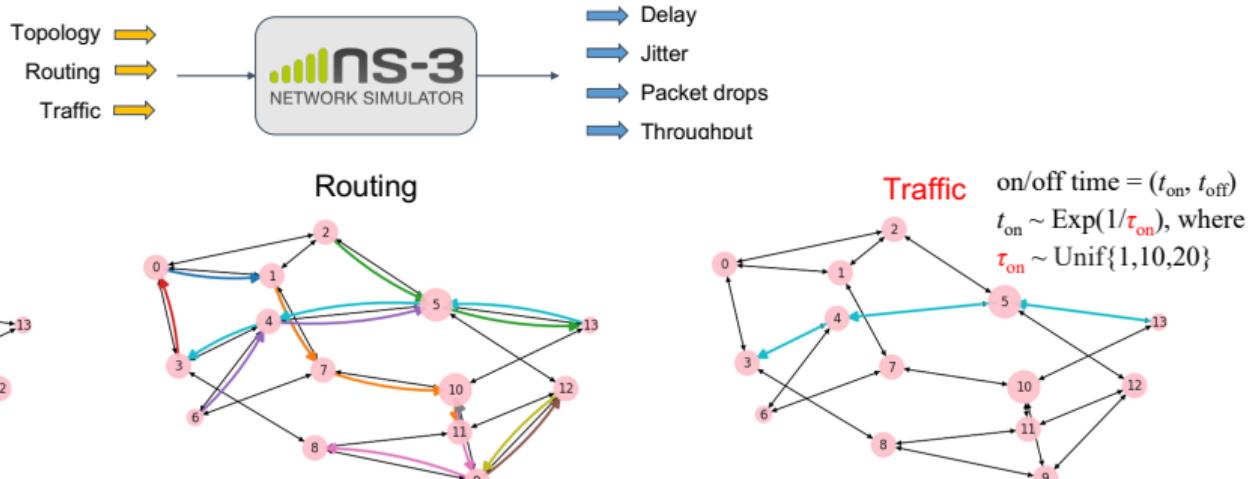
⁶B. Li, G. Verma, T. Efimov, A. Kumar, and S. Segarra, "GLANCE: Graph-based Learnable Digital Twin for Communication Networks", Preprint: arXiv:2408.09040

Introduction of network simulators



- ▶ Each **flow** corresponds to a set of KPIs (key performance indicators)
 - ⇒ Guide the design, evaluation, and optimization of networks & protocols

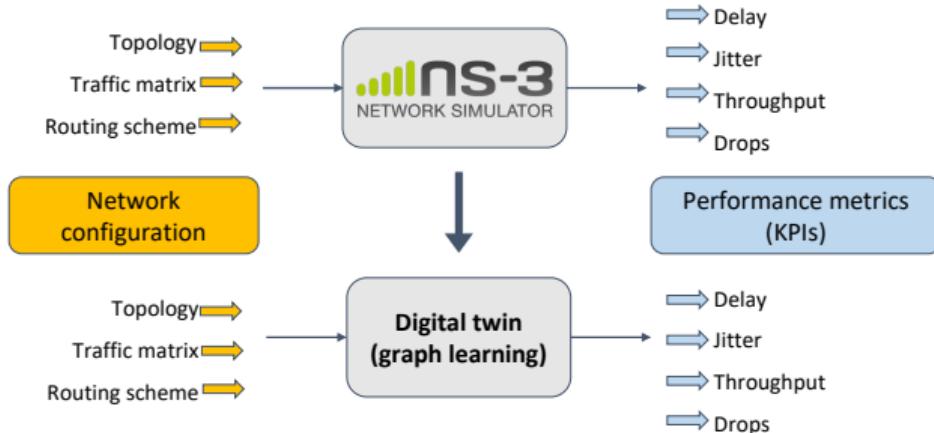
Introduction of network simulators



Example inputs: NSFNet (14 nodes, 42 links, 10 flows/paths)

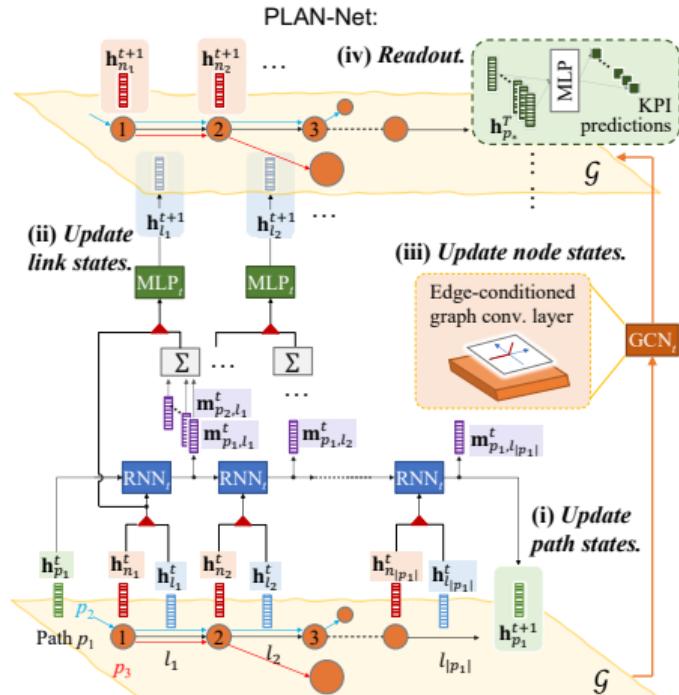
- ▶ Each **flow** corresponds to a set of KPIs (key performance indicators)
 - ⇒ Guide the design, evaluation, and optimization of networks & protocols
- ▶ Network simulator emulates every step in the network protocols and wireless channels
 - ⇒ Very **slow**, difficult to **scale up**

What network digital twin can do?



- ▶ Fast KPI prediction and differentiable process
- ▶ Digital twin of network simulators
 - ⇒ Predict KPIs rapidly (fast execution)
 - ⇒ Enable iterative optimization (fast execution)
 - ⇒ Training machine learning-based network solutions (differentiability)

PLAN-Net: message-passing architecture



* MLP: Multi-Layer Perceptron
 RNN: Recurrent Neural Network
 GCN: Graph Convolutional Network

Algorithm 1 PLAN-Net algorithm.

Input: Graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, routing list \mathcal{R}

Initialize: $\rho_p, c_l, d_n > 0$

1: $\mathbf{h}_p^0 \leftarrow [\rho_p, 0, \dots, 0]^\top, \forall p \in \mathcal{P}$

2: $\mathbf{h}_l^0 \leftarrow [c_l, 0, \dots, 0]^\top, \forall l \in \mathcal{L}$

3: $\mathbf{h}_n^0 \leftarrow [d_n, 0, \dots, 0]^\top, \forall n \in \mathcal{N}$

4: **for** $t = 0, 1, \dots, T - 1$ **do**

5: **(i) Update path states.**

6: **for** every path p in \mathcal{P} **do**

7: **for** every link l in p **do**

8: $\mathbf{h}_p^t \leftarrow \text{RNN}_t(\mathbf{h}_p^t, \text{cat}[\mathbf{h}_l^t, \mathbf{h}_n^t])$, where $n = \text{src}(l)$
 9: $\triangleright n$ is the source node of l

10: $\mathbf{m}_{p,l}^t \leftarrow \mathbf{h}_p^t$

11: **end for**

12: $\mathbf{h}_p^{t+1} \leftarrow \mathbf{h}_p^t$

13: **end for**

14: **(ii) Update link states.**

15: **for** every link l in \mathcal{L} **do**

16: $\mathbf{h}_l^{t+1} \leftarrow \text{MLP}_t(\text{cat}[\mathbf{h}_l^t, \mathbf{h}_n^t, \text{agg}\{\mathbf{m}_{p,l}^t \mid l \in p\}])$
 17: $\triangleright p$ is all paths that contain l

18: **end for**

19: **(iii) Update node states.**

20: **for** every node n in \mathcal{G} **do**

21: $\mathbf{h}_n^{t+1} \leftarrow \text{GCN}_t(\text{cat}[\mathbf{h}_n^t, \text{agg}\{\mathbf{h}_l^t \mid l = L^+(n)\}]; \mathcal{G})$
 22: $\triangleright l$ is all links out of n

23: **end for**

24: **end for**

25: **(iv) Readout.**

26: $\mathbf{y} = \text{MLP}(\mathbf{h}_p^{T+1})$

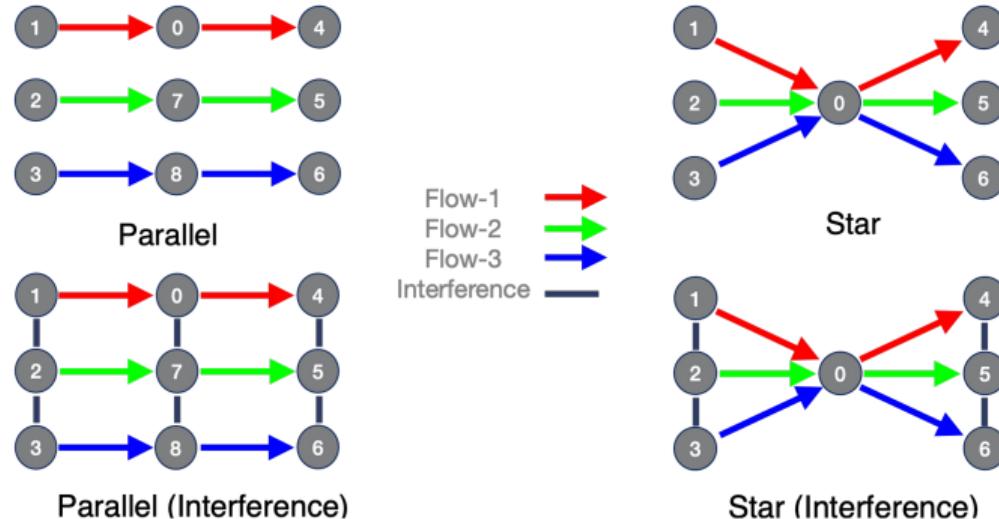
RNN

MLP

GCN

PLAN-Net (Path, Link, And Node)

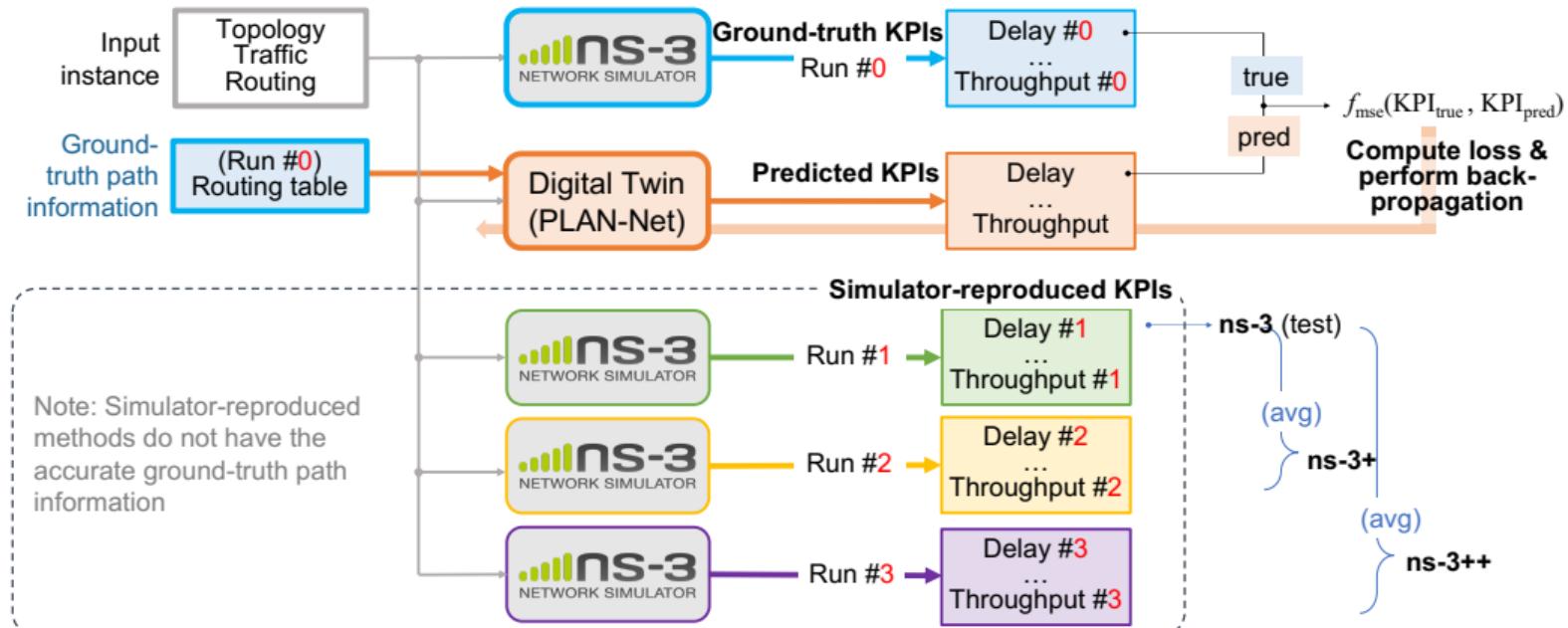
- ▶ PLAN-Net improves existing RouteNet⁷ for wired networks
- ▶ Leverage node embeddings to distinguish different interference topologies



⁷K. Rusek, et al., "RouteNet: Leveraging graph neural networks for network modeling and optimization in SDN," IEEE J. Sel. Areas Commun., vol. 38, no. 10, pp. 2260–2270, 2020.

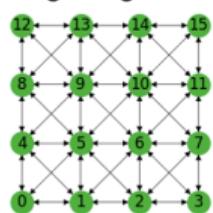
Training and evaluation

- ▶ Supervised training, using ns-3 single-run output as training labels
- ▶ Performance evaluated by mean absolute error (MAE)

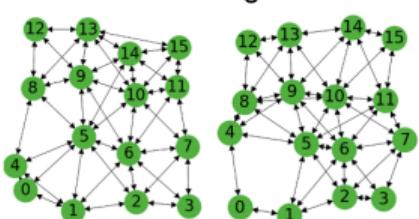


Tests: wireless networks of grid topology

Regular grid



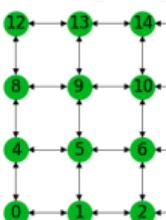
Perturbed grids



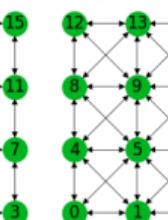
PLAN-Net is more accurate than a single run of ns-3

- ▶ Alter **transmit power** to test for different levels of interference
- ▶ PLAN-Net achieves the lowest MAE
- ▶ Generalize to different topologies
- ▶ PLAN-Net runs **1000x faster** than **ns-3**, e.g., 100s → 0.01-0.1 s

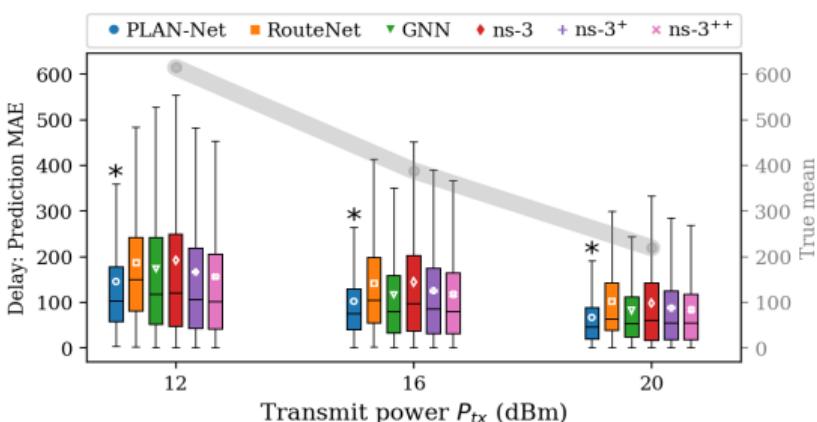
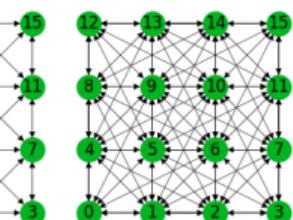
$P_{tx} = 12$



$P_{tx} = 16$ (default)



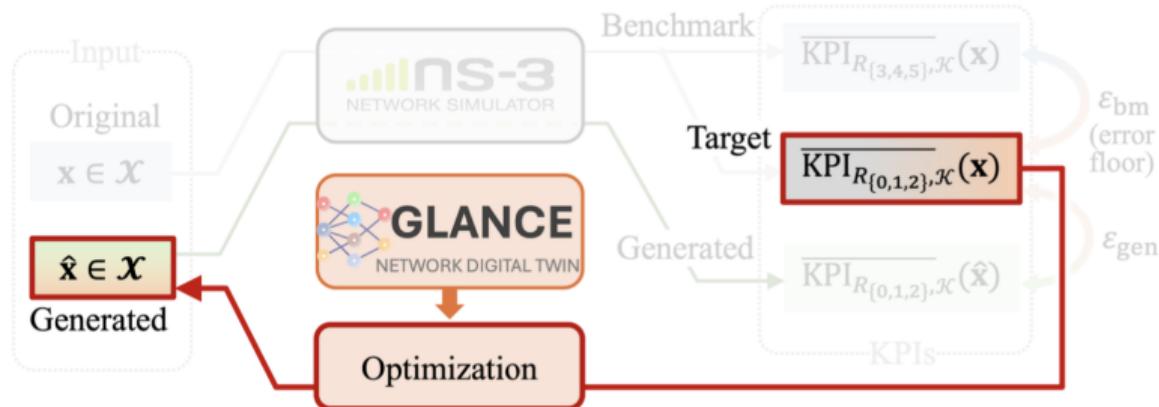
$P_{tx} = 20$



Managing Networks with Digital Twins

Problem definition

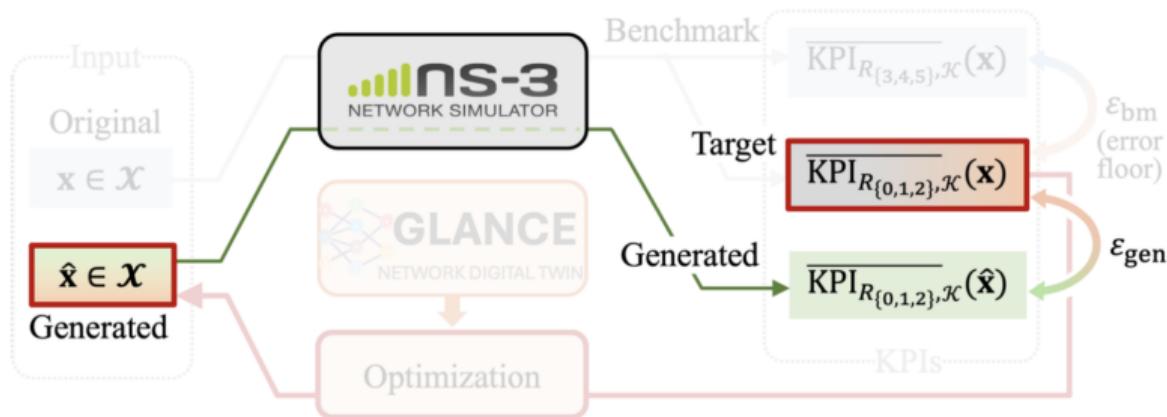
Given a **target KPI** profile, find a **network configuration** s.t. ...



Managing Networks with Digital Twins

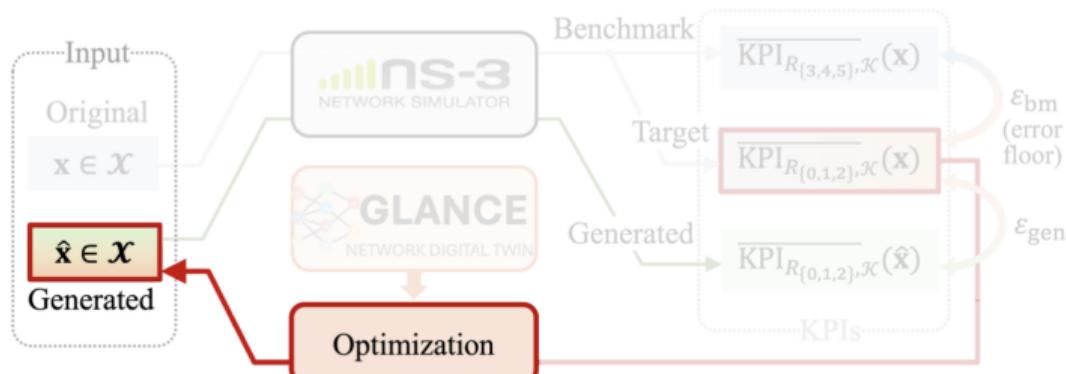
Problem definition

Given a **target KPI** profile, find a **network configuration** s.t. the resulting KPIs are close to the target.



Managing Networks with Digital Twins

Require different optimization approaches:



1. Load management

- Topology, flows, **traffic**
- Managing on/off parameters (continuous and differentiable)
- **Gradient-based optimization**

2. Traffic offloading

- Topology, **flows**, traffic
- Managing dst. node indices (discrete and non-differentiable)
- **Local search optimization**

Managing Networks with Digital Twins

Comparing ε_{gen} vs ε_{bm} in KPI space:

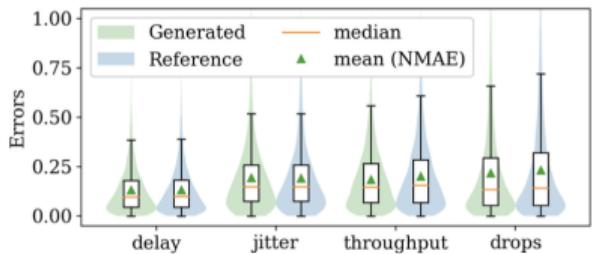


Figure: 1. Load management

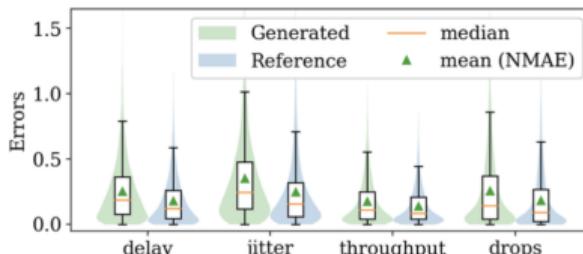
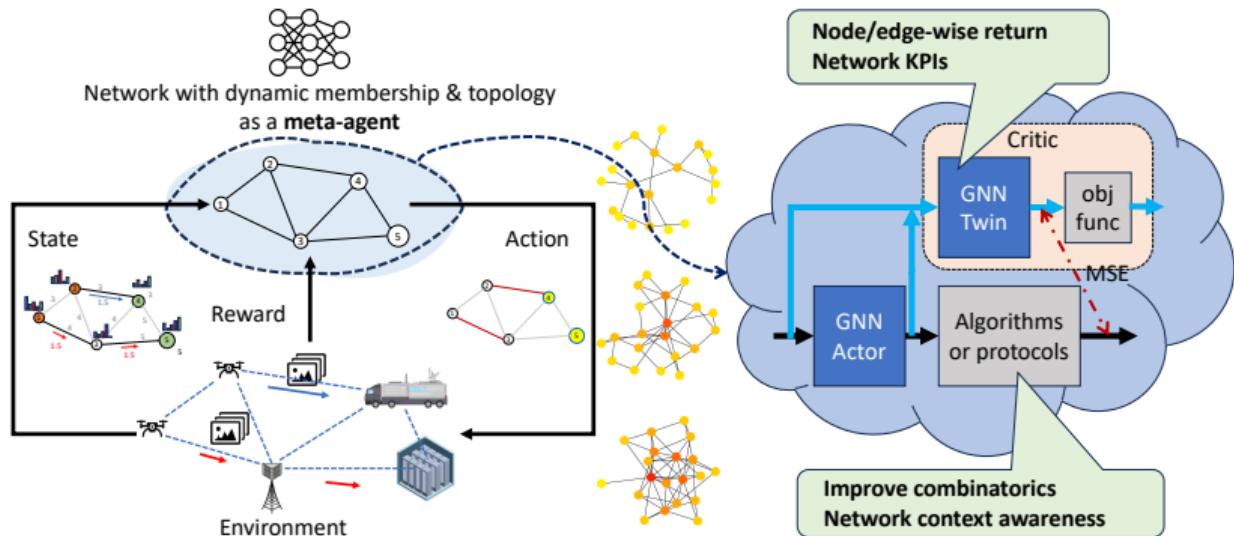


Figure: 2. Traffic offloading

Takeaways

- Note the difference in the gap between ε_{gen} and ε_{bm} heights
- GLANCE gradient-based management of traffic loads seems easier than local search-based management of traffic dst.

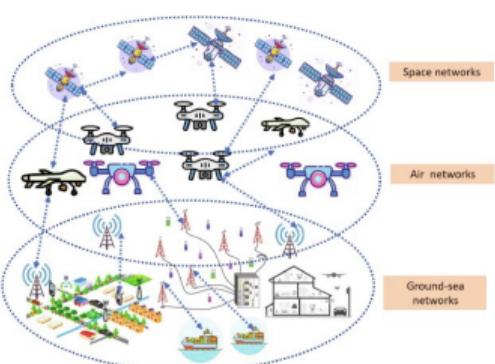
Part 3: Key Take Aways



- ▶ GNNs: **permutation invariant, distributed exec.** (scalability),
 ⇒ Dynamic network topology & membership, parallel nature
- ▶ Hybrid ML pipelines → domain knowledge as **inductive bias**
- ▶ Digital twins → fast & differentiable simulators as new optimization tools

Potential Applications in Networked Systems

5G/6G networks



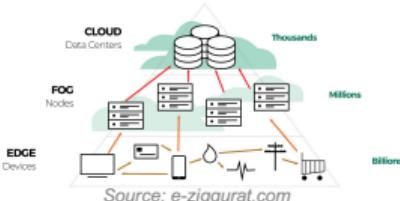
Ad-hoc & sensor networks



Smart traffic light control system



Cloud, fog, edge computing



Operations Research



Conclusions and Future Directions

Going back to our Key Takeaways

- ▶ What are graph neural networks (GNNs)?
- ▶ Why are GNNs well suited to tackle problems in wireless communications?
- ▶ How have GNNs been applied to specific problems?
- ▶ What are open problems/challenges to which you can contribute?

Going back to our Key Takeaways

- ▶ Class of **parametric, layered, non-linear** functions that incorporate information both from **features** and **graph structure**
- ▶ Why are GNNs well suited to tackle problems in wireless communications?
- ▶ How have GNNs been applied to specific problems?
- ▶ What are open problems/challenges to which you can contribute?

Going back to our Key Takeaways

- ▶ Class of **parametric, layered, non-linear** functions that incorporate information both from **features** and **graph structure**
- ▶ **Scalability, distributed** implementation, and **permutation equivariance/invariance**
- ▶ How have GNNs been applied to specific problems?
- ▶ What are open problems/challenges to which you can contribute?

Going back to our Key Takeaways

- ▶ Class of **parametric, layered, non-linear** functions that incorporate information both from **features** and **graph structure**
- ▶ **Scalability, distributed** implementation, and **permutation equivariance/invariance**
- ▶ We covered **power allocation & beamforming** and **link scheduling**
- ▶ What are open problems/challenges to which you can contribute?

Going back to our Key Takeaways

- ▶ Class of **parametric, layered, non-linear** functions that incorporate information both from **features** and **graph structure**
- ▶ **Scalability, distributed** implementation, and **permutation equivariance/invariance**
- ▶ We covered **power allocation & beamforming** and **link scheduling**
- ▶ Hopefully, the technical discussion have triggered some thoughts. We will also discuss **open directions** now

Open and Future Directions

- ▶ The ‘easy’ one \Rightarrow Applications to other problems in wireless (and beyond)

Open and Future Directions

- ▶ The ‘easy’ one ⇒ Applications to other problems in wireless (and beyond)
- ▶ Implementation in real wireless networks
 - ⇒ Fading, inexact channel info, packet drops, adversarial/malfunctioning nodes
 - ⇒ Specific protocols for message passing implementation

Open and Future Directions

- ▶ The ‘easy’ one ⇒ **Applications** to other problems in wireless (and beyond)
- ▶ Implementation in **real wireless networks**
 - ⇒ Fading, inexact channel info, packet drops, adversarial/malfunctioning nodes
 - ⇒ Specific protocols for message passing implementation
- ▶ Efficient **distributed training**
 - ⇒ New challenges in distributed optimization
 - ⇒ Key for implementation in real systems

Open and Future Directions

- ▶ The ‘easy’ one ⇒ **Applications** to other problems in wireless (and beyond)
- ▶ Implementation in **real wireless networks**
 - ⇒ Fading, inexact channel info, packet drops, adversarial/malfunctioning nodes
 - ⇒ Specific protocols for message passing implementation
- ▶ Efficient **distributed training**
 - ⇒ New challenges in distributed optimization
 - ⇒ Key for implementation in real systems
- ▶ Combination with **generative AI**
 - ⇒ Data augmentation and large training datasets

Open and Future Directions

- ▶ The ‘easy’ one ⇒ Applications to other problems in wireless (and beyond)
- ▶ Implementation in real wireless networks
 - ⇒ Fading, inexact channel info, packet drops, adversarial/malfunctioning nodes
 - ⇒ Specific protocols for message passing implementation
- ▶ Efficient distributed training
 - ⇒ New challenges in distributed optimization
 - ⇒ Key for implementation in real systems
- ▶ Combination with generative AI
 - ⇒ Data augmentation and large training datasets
- ▶ Privacy-preserving message passing in GNNs

Open and Future Directions

- ▶ The ‘easy’ one ⇒ **Applications** to other problems in wireless (and beyond)
- ▶ Implementation in **real wireless networks**
 - ⇒ Fading, inexact channel info, packet drops, adversarial/malfunctioning nodes
 - ⇒ Specific protocols for message passing implementation
- ▶ Efficient **distributed training**
 - ⇒ New challenges in distributed optimization
 - ⇒ Key for implementation in real systems
- ▶ Combination with **generative AI**
 - ⇒ Data augmentation and large training datasets
- ▶ **Privacy-preserving** message passing in GNNs
- ▶ **Uncertainty** and implementation in critical infrastructure

Thank you

Santiago Segarra (segarra@rice.edu)

Ananthram Swami (ananthram.swami.civ@army.mil)

Zhongyuan Zhao (zhongyuan.zhao@rice.edu)

Related Publications

1. Eli Chien, Mufei Li, Anthony Aportela, Kerr Ding, Shuyi Jia, Supriyo Maji, Zhongyuan Zhao, Victor Fung, Callie Hao, Yunan Luo, Olgica Milenkovic, David Pan, Santiago Segarra, Javier Duarte, and Pan Li. "Exploring the opportunities and challenges of graph neural networks in electrical engineering," *Nature Reviews Electrical Engineering*, 2024, (To appear).
2. A. Chowdhury, G. Verma, C. Rao, A. Swami and S. Segarra, "Unfolding WMMSE Using Graph Neural Networks for Efficient Power Allocation," in *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 6004-6017, Sept. 2021.
3. A. Chowdhury, G. Verma, A. Swami and S. Segarra, "Deep Graph Unfolding for Beamforming in MU-MIMO Interference Networks," in *IEEE Transactions on Wireless Communications*, Oct. 2023
4. A. Chowdhury, G. Verma, C. Rao, A. Swami and S. Segarra, "Efficient Power Allocation Using Graph Neural Networks and Deep Algorithm Unfolding," *IEEE ICASSP*, Toronto, ON, Canada, 2021, pp. 4725-4729
5. A. Chowdhury, S. Paternain, G. Verma, A. Swami and S. Segarra, "Learning Non-myopic Power Allocation in Constrained Scenarios," *2023 57th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, 2023, pp. 804-808.
6. B. Li, J. Perazzone, A. Swami and S. Segarra, "Learning to Transmit with Provable Guarantees in Wireless Federated Learning," in *IEEE Transactions on Wireless Communications*, Dec. 2023
7. B. Li, A. Swami and S. Segarra, "Power Allocation for Wireless Federated Learning Using Graph Neural Networks," *IEEE ICASSP*, Singapore, Singapore, 2022, pp. 5243-5247

Related Publications (continue)

8. Z. Zhao, G. Verma, C. Rao, A. Swami and S. Segarra, "Distributed Scheduling Using Graph Neural Networks," IEEE ICASSP, Toronto, ON, Canada, 2021, pp. 4720-4724.
9. Z. Zhao, G. Verma, C. Rao, A. Swami and S. Segarra, "Link Scheduling Using Graph Neural Networks," in IEEE Trans. on Wireless Communications, vol. 22, no. 6, pp. 3997-4012, 2023
10. Z. Zhao, A. Swami, S. Segarra, "Graph-based Deterministic Policy Gradient for Repetitive Combinatorial Optimization Problems," ICLR 2023
11. Z. Zhao, G. Verma, A. Swami and S. Segarra, "Delay-Oriented Distributed Scheduling Using Graph Neural Networks," IEEE ICASSP, Singapore, Singapore, 2022, pp. 8902-8906
12. Z. Zhao, B. Radojicic, G. Verma, A. Swami and S. Segarra, "Delay-Aware Backpressure Routing Using Graph Neural Networks," IEEE ICASSP, Rhodes Island, Greece, 2023, pp. 1-5
13. Z. Zhao, G. Verma, A. Swami and S. Segarra, "Enhanced Backpressure Routing Using Wireless Link Features," IEEE CAMSAP, Herradura, Costa Rica, 2023, pp. 271-275
14. Z. Zhao, B. Radojičić, G. Verma, A. Swami, S. Segarra, "Biased Backpressure Routing Using Link Features and Graph Neural Networks," submitted to IEEE Trans. on Machine Learning In Communications and Networking, (under review).
15. B. Li, T. Efimov, A. Kumar, J. Cortes, G. Verma, A. Swami, and S. Segarra. "Learnable Digital Twin for Efficient Wireless Network Evaluation." In IEEE MILCOM, pp. 661-666., 2023.
16. Z. Zhao, J. Perazzone, G. Verma and S. Segarra, "Congestion-Aware Distributed Task Offloading in Wireless Multi-Hop Networks Using Graph Neural Networks," IEEE ICASSP, Seoul, Republic of Korea, 2024, pp. 8951-8955.

Open Source Repositories

Physical layer: power control & MIMO

1. <https://github.com/ArCho48/Unrolled-WMMSE>
2. <https://github.com/ArCho48/Unrolled-WMMSE-for-MU-MIMO>
3. https://github.com/ArCho48/UWMMSE_MIMO
4. <https://github.com/ArCho48/stability-UWMMSE>
5. Power control for federated learning: https://github.com/bl166/usca_power_control
6. <https://github.com/bl166/WirelessFL-PDG>

Distributed combinatorial optimization

7. Link scheduling: <https://github.com/zhongyuanzhao/distgcn>
8. Delay-oriented link scheduling: <https://github.com/zhongyuanzhao/gcn-dql>
9. GPG-Twin <https://github.com/XzrTGMu/twin-nphard>

Biased Backpressure routing, Network Digital Twin, and Multihop Offloading

10. Biased Backpressure routing: <https://github.com/zhongyuanzhao/biasBP> (to appear soon)
11. Network Digital Twin: https://github.com/bl166/wireless_digital_twin_milcom
12. Multihop Offloading: <https://github.com/zhongyuanzhao/multihop-offload>

Python Libraries

Deep Learning Platform



Platform-agnostic

Graph Neural Networks



Build your models with PyTorch,
TensorFlow or Apache MXNet.

Graphs, Networks,
Topological Domains



NetworkX is a Python package for
the creation, manipulation, and
study of the structure, dynamics,
and functions of complex networks.



<https://pyt-team.github.io/>

TopoX are a set of Python packages
for meeting diverse computational
needs on topological domains.