Graph-based Machine Learning for Wireless Communications

Santiago Segarra*, Ananthram Swami‡, Zhongyuan Zhao*

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

May 5th, 2024

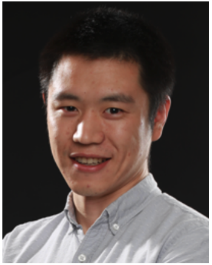
IEEE ICMLCN 2024
Stockholm, Sweden

Slides PDF

The team

Santiago Segarra
ECE, Rice

Ananthram Swami
DEVCOM, ARL



Zhongyuan Zhao
ECE, Rice

DEVCOM ARMY RESEARCH LABORATORY

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

Acknowledgements

- Rice University
  - ⇒ Arindam Chowdhury
  - ⇒ Boning Li
  - ⇒ Ashu Sabharwal
- Army Research Lab
  - ⇒ Gunjan Verma
  - ⇒ Jake Perazzone
  - ⇒ Kevin Chan
  - ⇒ Chirag Rao

Key Takeaways

By 5:30 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

Part III: Graph-based ML for Wireless Networking

a) Introduction to networking tasks
b) Link scheduling
c) Graph-based actor-critic reinforcement learning framework
d) GNNs for Backpressure Routing
e) Digital twin of wireless networks

Conclusions and Future Directions

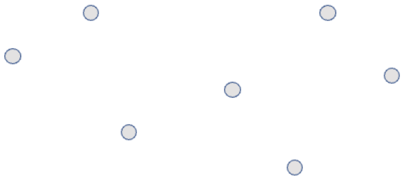
Part I: Introduction to Graph Neural Networks

Graph-based Machine Learning

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

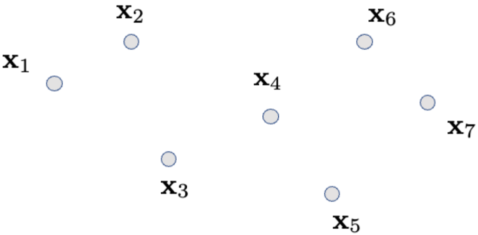
Graph-based Machine Learning

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



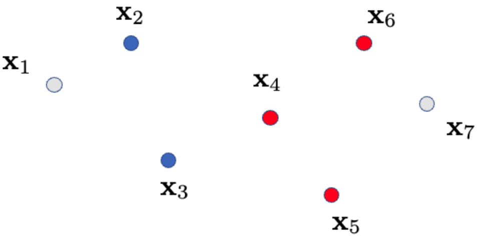
Graph-based Machine Learning

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

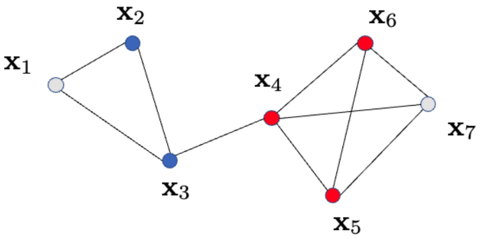


- Classical supervised learning setting

  $\Rightarrow$ Learn a parametric function that estimates the labels $\Rightarrow \hat{y}_i = f_{\boldsymbol{\theta}}(\mathbf{x}_i)$

Graph-based Machine Learning

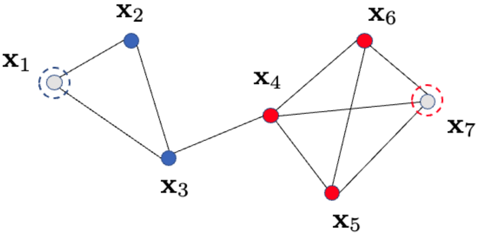
Graph-based ML leverages the network structure of the data to improve learning and processing of these data



- 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

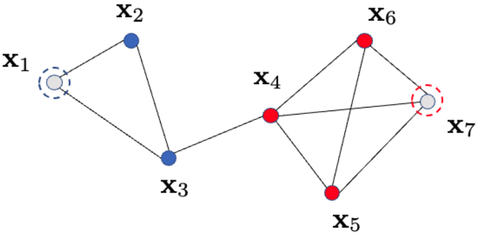


- The structure also carries information about node labels

  ⇒ Estimate labels by combining both node features and graph structure

Graph-based Machine Learning

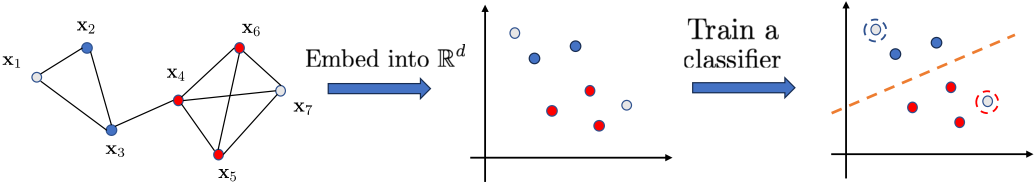
Graph-based ML leverages the network structure of the data to improve learning and processing of these data



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

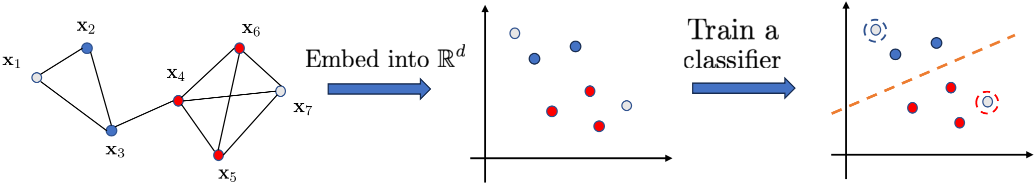
Graph Representation Learning

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



Graph Representation Learning

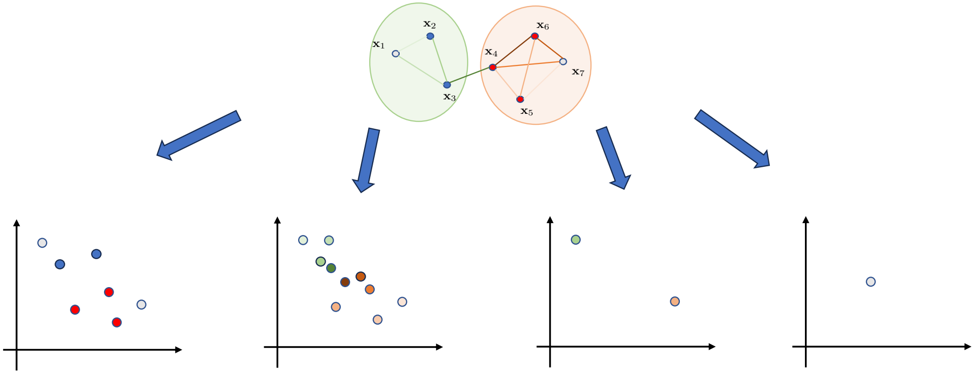
- Convert raw graph data into a low-dimensional vector representation



- Once in $\mathbb{R}^d$, we can apply the whole ML machinery
- Embedding can be unsupervised
  - $\Rightarrow$ "Closeness" in the graph is preserved as "closeness" in $\mathbb{R}^d$
- or supervised
  - $\Rightarrow$ 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



Embed nodes → Node classification / regression

Embed edges (or node pairs) → Edge classification / regression
Link prediction

Embed subgraphs → Subgraph "function" discovery

Embed graphs → Graph classification / regression

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

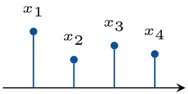
Graph Representation Learning

- "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

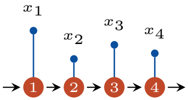
Graph-structured data

- Discrete-time signal $\Rightarrow$ Relation of nearby values carries information



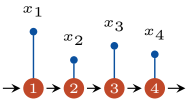
Graph-structured data

- Discrete-time signal $\Rightarrow$ Relation of nearby values carries information
  $\Rightarrow$ Make the data structure explicit $\Rightarrow$ Nearby elements are related
  $\Rightarrow$ Two constitutive elements of SP: data structure and signal values

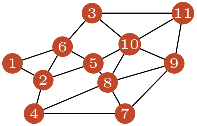


Graph-structured data

- Discrete-time signal $\Rightarrow$ Relation of nearby values carries information
  $\Rightarrow$ Make the data structure explicit $\Rightarrow$ Nearby elements are related
  $\Rightarrow$ Two constitutive elements of SP: data structure and signal values

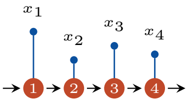


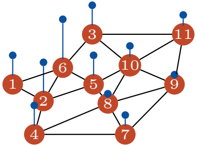
- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ $\Rightarrow$ $\mathcal{V}$: set of nodes, $\mathcal{E}$: set of edges



Graph-structured data

- Discrete-time signal $\Rightarrow$ Relation of nearby values carries information
  - $\Rightarrow$ Make the data structure explicit $\Rightarrow$ Nearby elements are related
  - $\Rightarrow$ Two constitutive elements of SP: data structure and signal values
- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ $\Rightarrow$ $\mathcal{V}$: set of nodes, $\mathcal{E}$: set of edges
- Graph signals $\Rightarrow$ Associate a value to each node $\mathsf{x} : \mathcal{V} \to \mathbb{R}$

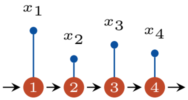


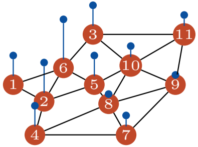


Graph-structured data

- Discrete-time signal $\Rightarrow$ Relation of nearby values carries information
  - $\Rightarrow$ Make the data structure explicit $\Rightarrow$ Nearby elements are related
  - $\Rightarrow$ Two constitutive elements of SP: data structure and signal values
- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ $\Rightarrow$ $\mathcal{V}$: set of nodes, $\mathcal{E}$: set of edges
- Graph signals $\Rightarrow$ Associate a value to each node $\mathsf{x} : \mathcal{V} \to \mathbb{R}$
- Matrix representation $\Rightarrow$ Adjacency matrix $\mathbf{A}$

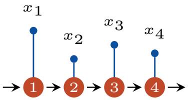
$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$



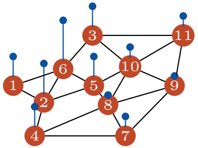


Graph-structured data

- Discrete-time signal $\Rightarrow$ Relation of nearby values carries information
  $\Rightarrow$ Make the data structure explicit $\Rightarrow$ Nearby elements are related
  $\Rightarrow$ Two constitutive elements of SP: data structure and signal values



- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ $\Rightarrow$ $\mathcal{V}$: set of nodes, $\mathcal{E}$: set of edges
- Graph signals $\Rightarrow$ Associate a value to each node $\mathsf{x} : \mathcal{V} \to \mathbb{R}$
- Matrix representation $\Rightarrow$ Adjacency matrix $\mathbf{A}$

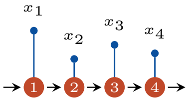


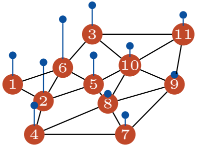
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Graph-structured data

- Discrete-time signal $\Rightarrow$ Relation of nearby values carries information
  - $\Rightarrow$ Make the data structure explicit $\Rightarrow$ Nearby elements are related
  - $\Rightarrow$ Two constitutive elements of SP: data structure and signal values
- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ $\Rightarrow$ $\mathcal{V}$: set of nodes, $\mathcal{E}$: set of edges
- Graph signals $\Rightarrow$ Associate a value to each node $\mathsf{x} : \mathcal{V} \to \mathbb{R}$
- Matrix representation $\Rightarrow$ Adjacency matrix $\mathbf{A}$, Laplacian matrix $\mathbf{L}$

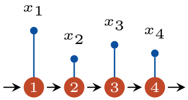
$$\mathbf{L} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & 3 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & -1 & 0 \\ -1 & -1 & -1 & 0 & -1 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 3 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & -1 & 5 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & -1 & 0 & -1 & 0 & 0 & -1 & -1 & 5 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 3 \end{bmatrix}$$

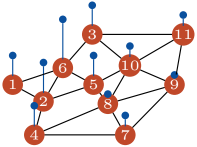




Graph-structured data

- Discrete-time signal $\Rightarrow$ Relation of nearby values carries information
  - $\Rightarrow$ Make the data structure explicit $\Rightarrow$ Nearby elements are related
  - $\Rightarrow$ Two constitutive elements of SP: data structure and signal values
- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ $\Rightarrow$ $\mathcal{V}$: set of nodes, $\mathcal{E}$: set of edges
- Graph signals $\Rightarrow$ Associate a value to each node $\mathrm{x} : \mathcal{V} \to \mathbb{R}$
- Matrix representation $\Rightarrow$ Adjacency matrix $\mathbf{A}$, Laplacian matrix $\mathbf{L}$
  - $\Rightarrow$ Fixes ordering of the nodes $\Rightarrow$ Permutations
  - $\Rightarrow$ Generic matrix $\mathbf{S}$ (support matrix, graph shift operator)





Sandryhaila, Moura, "Discrete Signal Processing on Graphs", IEEE TSP, 2013

Shuman, Narang, Frossard, Ortega, Vandergheynst, "The Emerging Field of Signal Processing on Graphs", IEEE SPM, 2013

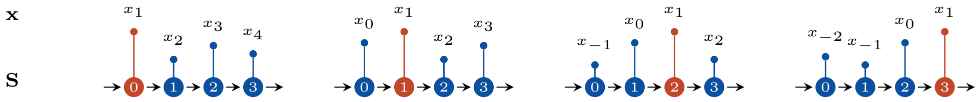
Graph convolutions

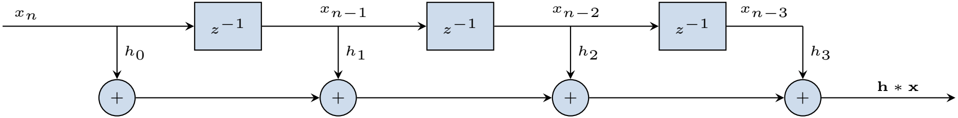
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal

$$\mathbf{x} * \mathbf{h} = \sum_{k=0}^{K-1} h_k x_{n-k}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph (adjacency, Laplacian)

$$\begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}$$



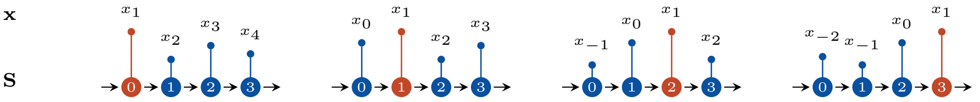


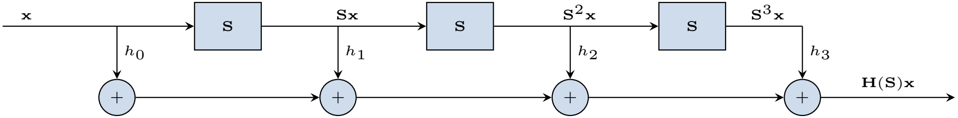
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal

$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{Sx}$ shifts the signal $\mathbf{x}$

$$\begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix}$$





Graph convolutions

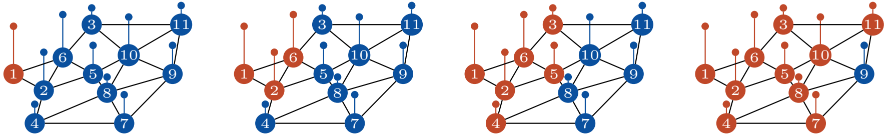
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal

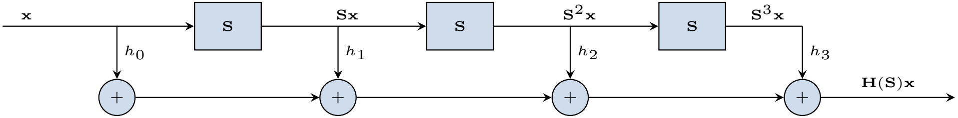
$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{S}\mathbf{x}$ shifts the signal $\mathbf{x}$

$$\begin{bmatrix} 0&1&0&0&0&1&0&0&0&0&0 \\ 1&0&0&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&1 \\ 0&1&0&0&0&0&1&1&0&0&0 \\ 0&1&0&0&1&0&0&1&0&1&0 \\ 1&1&1&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&1&0&0 \\ 0&0&0&1&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&1&1&0&1&1 \\ 0&0&0&0&1&0&0&1&1&0&1 \\ 0&0&1&0&0&0&0&0&1&1&0 \end{bmatrix}$$

$(\mathbf{x}; \mathbf{S})$





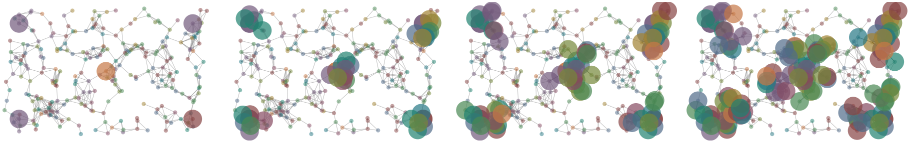
Graph convolutions

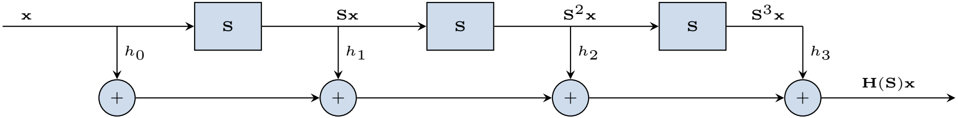
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal

$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph (adjacency, Laplacian)
- Linear combination of neighboring signal $\Rightarrow$ Local operation

$(\mathbf{x}; \mathbf{S})$





Nonlinear graph signal processing

Traditional signal processing

$\Rightarrow$ Best linear filter that exploits structure

$$\min_{\{h_k\}} \mathsf{J}(\mathbf{z}_1) = \min_{\{h_k\}} \mathsf{J}(\mathbf{H}(\mathbf{S})\mathbf{x})$$

Linear models $\Rightarrow$ Limited representation

$\Rightarrow$ Nonlinear graph signal processing

$\mathbf{x}$

$$\mathbf{z}_1 = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}$$

$\mathbf{z}_1$

Gama, Isufi, Leus, Ribeiro, “Graphs, Convolutions and Neural Networks: From Graph Filters to Graph Neural Networks”, IEEE SPM, 2020

Nonlinear graph signal processing

Traditional signal processing

$\Rightarrow$ Best linear filter that exploits structure

$$\min_{\{h_k\}} \mathsf{J}(\mathbf{z}_1) = \min_{\{h_k\}} \mathsf{J}(\mathbf{H}(\mathbf{S})\mathbf{x})$$

Linear models $\Rightarrow$ Limited representation

$\Rightarrow$ Nonlinear graph signal processing

Graph perceptron $\Rightarrow$ Nonlinear processing

$\Rightarrow$ Graph filter $\Rightarrow$ Pointwise nonlinearity

$\Rightarrow$ *Learn* graph filter $\{h_k\}$ $\Rightarrow$ $\min_{\{h_k\}} J(\mathbf{x}_1)$

Basic nonlinear description of models

$\Rightarrow$ Increase representation power $\Rightarrow$ Repeat

$$\mathbf{x} \rightarrow \boxed{\mathbf{z}_1 = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}} \xrightarrow{\mathbf{z}_1} \boxed{\mathbf{x}_1 = \sigma\left[\mathbf{z}_1\right]} \rightarrow \mathbf{x}_1$$

Gama, Isufi, Leus, Ribeiro, "Graphs, Convolutions and Neural Networks: From Graph Filters to Graph Neural Networks", IEEE SPM, 2020

Graph convolutional networks

Cascade of L layers

$\Rightarrow$ Graph convolutions with filters $\mathcal{H} = \{\mathbf{h}_\ell\}$

$\Rightarrow$ Pointwise nonlinearity (activation functions)

The GCNN $\boldsymbol{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$ depends on the filters $\mathcal{H}$

$\Rightarrow$ Learn filter taps $\mathcal{H}$ from training data

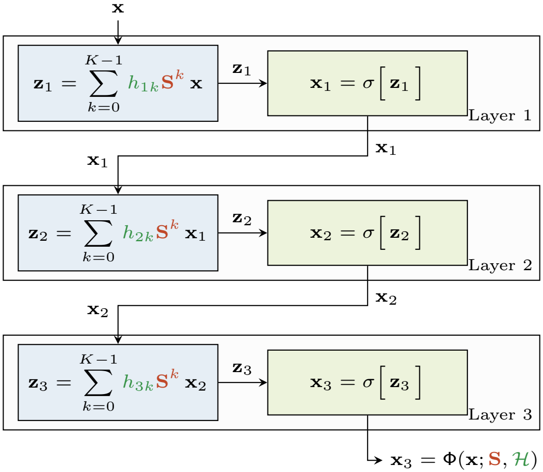
$\Rightarrow$ Also depends on the graph $\mathbf{S}$

Nonlinear mapping $\boldsymbol{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$

$\Rightarrow$ Exploit underlying graph structure $\mathbf{S}$

$\Rightarrow$ Local information

$\Rightarrow$ Distributed implementation

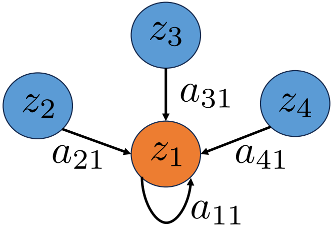


Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP, 2018

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)



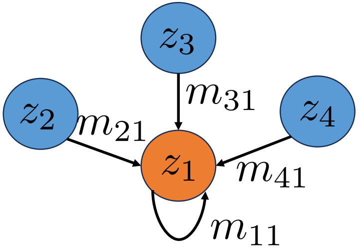
$$z_i^{(0)} = x_i$$

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

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)

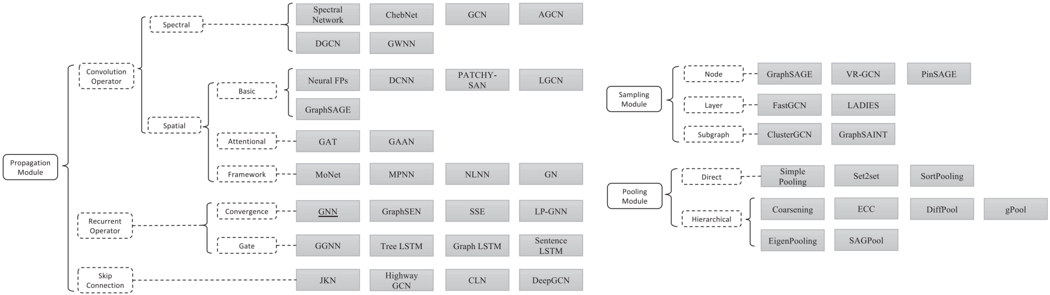


$$z_i^{(0)} = x_i$$

$$m_{ji}^{(k)} = f_{\mathrm{e}}(z_i^{(k)}, z_j^{(k)}, e_{ji})$$

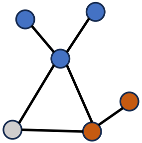
$$z_i^{(k+1)} = f_{\mathrm{v}}\left(z_i^{(k)}, \sum_{j \in (\mathcal{N}_i \cup \{i\})} m_{ji}^{(k)} \right)$$

A Zoo of GNNs has been developed

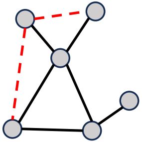


Credit: “Graph neural networks: A review of methods and applications”

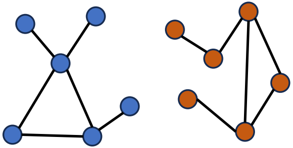
Typical generic problems tackled with GNNs



Node classification



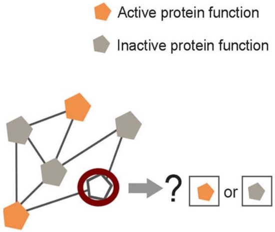
Link prediction



Graph classification

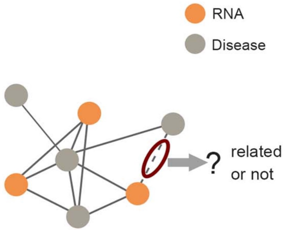
Translate into Domain-Specific Problems

Node classification



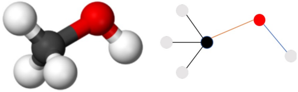
PPI Network

Link Prediction



RNA-Disease Association Network

Graph Classification

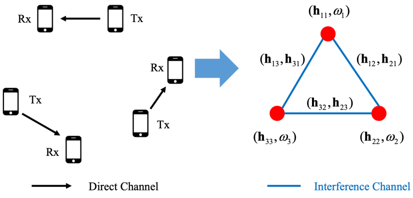


Soluble molecule or not

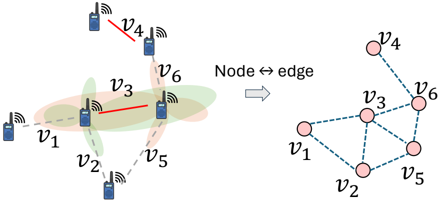
Credit: “Graph Neural Networks and their Current Applications in Bioinformatics”, Zhang et al.

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

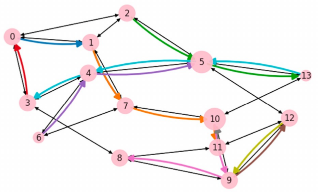
Graphs in Wireless Communications



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



Routing



Why GNNs for Wireless Communications?

- Built-in scalability

  $\Rightarrow$ We can train and test with different sizes of systems

Why GNNs for Wireless Communications?

- Built-in scalability
  - $\Rightarrow$ We can train and test with different sizes of systems
- Facilitate distributed implementation
  - $\Rightarrow$ 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?

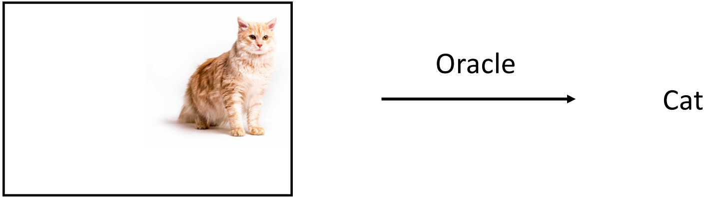
- Built-in scalability
  - $\Rightarrow$ We can train and test with different sizes of systems
- Facilitate distributed implementation
  - $\Rightarrow$ Forward-pass implementation based on local computations
- Locality plays a central role
  - $\Rightarrow$ My optimal decision depends on the parts of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

CNNs and translation invariance

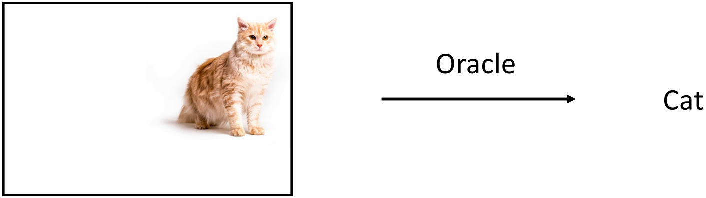
Oracle

Cat

CNNs and translation invariance



CNNs and translation invariance



- Architectures used for object recognition benefit from translation invariance
  $\Rightarrow$ Convolutional Neural Networks
- Learning in the class of function to which the oracle belongs

GNNs and permutation equivariance

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

GNNs and permutation equivariance

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

GNNs and permutation equivariance

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

- Architectures used for power allocation benefit from permutation equivariance
  $\Rightarrow$ Graph Neural Networks
- Learning in the class of function to which the oracle belongs

Permutation equivalence vs. invariance

- Equivariance $\Rightarrow f_{\boldsymbol{\theta}}(\mathbf{\Pi X}; \mathbf{\Pi A \Pi}^\top) = \mathbf{\Pi} f_{\boldsymbol{\theta}}(\mathbf{X}; \mathbf{A})$
- Invariance $\Rightarrow f_{\boldsymbol{\theta}}(\mathbf{\Pi X}; \mathbf{\Pi A \Pi}^\top) = f_{\boldsymbol{\theta}}(\mathbf{X}; \mathbf{A})$

Permutation equivalence vs. invariance

- Equivariance $\Rightarrow f_{\boldsymbol{\theta}}(\boldsymbol{\Pi}\mathbf{X}; \boldsymbol{\Pi}\mathbf{A}\boldsymbol{\Pi}^\top) = \boldsymbol{\Pi} f_{\boldsymbol{\theta}}(\mathbf{X}; \mathbf{A})$
- Invariance $\Rightarrow f_{\boldsymbol{\theta}}(\boldsymbol{\Pi}\mathbf{X}; \boldsymbol{\Pi}\mathbf{A}\boldsymbol{\Pi}^\top) = f_{\boldsymbol{\theta}}(\mathbf{X}; \mathbf{A})$
- GNNs are equivariant at the level of the nodes (or edges) and invariant at the level of the graph
  - $\Rightarrow$ Node labels permute when the input is permuted
  - $\Rightarrow$ Graph labels are impervious to permutations
- Achievable rates (node-level quantity) are re-indexed with permutations
  - $\Rightarrow$ but the total sum-rate (graph-level quantity) is not modified

Model-inspired Data-driven Solutions

Theoretical model

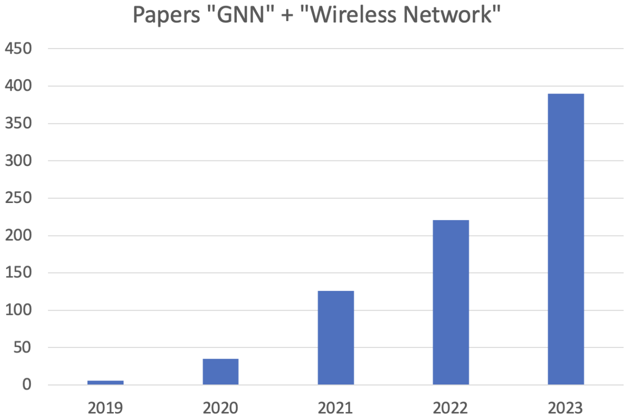
Accurate	Tractable		
✓	✓	Point-to-point channel capacity	Model-based
✓	✗	Sum-rate in interference-limited systems	Model-inspired data-driven solutions
✗	✓	Energy consumption, hardware impairment	
✗	✗	Molecular communications	Data-driven

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



Overview of GNN applications to Wireless Comms

- Very dynamic field $\Rightarrow$ Many new papers being published
- Several tutorials/surveys in the area

  $\Rightarrow$ He et al., "An overview on the application of graph neural networks in wireless networks", IEEE O. J. of the Comm. Soc., 2021

  $\Rightarrow$ Hu et al., "Distributed Machine Learning for Wireless Communication Networks: Techniques, Architectures, and Applications", IEEE Comm. Surv. & Tut., 2021

  $\Rightarrow$ Shen et al., "Graph neural networks for wireless communications: From theory to practice", IEEE Trans. Wireless Comm., 2022

  $\Rightarrow$ 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 $\Rightarrow$ Many new papers being published
- Several tutorials/surveys in the area

$\Rightarrow$ Simeone, "A Very Brief Introduction to Machine Learning with Applications to Communication Systems", IEEE Trans. on Cognitive Comm. and Netw., 2018

$\Rightarrow$ Ahmad et al., "Machine Learning Meets Communication Networks: Current Trends and Future Challenges", IEEE Access, 2020

$\Rightarrow$ Ali et al., "6G White Paper on Machine Learning in Wireless Communication Networks", Arxiv, 2020

$\Rightarrow$ Jiang, "Graph-based deep learning for communication networks: A survey", Computer Comm., 2022

$\Rightarrow$ 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

Motivation

- Power and bandwidth are fundamental resources in communication
  - $\Rightarrow$ Key to determine the effective capacity of a wireless network

Motivation

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

Motivation

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

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

Optimal Power Allocation - SISO Case

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 $\mathbf{H}(t) \in \mathbb{R}^{m \times m}$
  - $\Rightarrow$ Encodes (time-varying) channel characteristics
  - $\Rightarrow$ $H_{ji}(t)$ represents the channel from Tx i to Rx $r(j)$ at time t

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

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

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 $\mathbf{H}(t) \in \mathbb{R}^{m \times m}$
  - $\Rightarrow$ Encodes (time-varying) channel characteristics
  - $\Rightarrow$ $H_{ji}(t)$ represents the channel from Tx i to Rx $r(j)$ at time t

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

  - $\Rightarrow$ where $H_{ji}^{P} \propto \mathrm{dist}(i, r(j))^{-k}$ and $H_{ji}^{F}(t) \sim \mathrm{Rayleigh}(\alpha)$
- Node State Information (NSI) matrix $\mathbf{X}(t) \in \mathbb{R}^{m \times d}$
  - $\Rightarrow$ Encodes (time-varying) node features of the Tx-Rx pair
  - $\Rightarrow$ # 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: sum rate across nodes

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: sum rate across nodes
- Data rate at receiver i is given by (for noise variance σ^2)

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

  - Maximize weighted sum-rate $\sum_{i=1}^m \alpha_i c_i$, under power constraint
  - 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]
  - $\Rightarrow$ Reformulate the optimization problem
  - $\Rightarrow$ Implement block coordinate descent
  - $\Rightarrow$ Leads to closed-form iteration formulas

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

Classical Approach

- Weighted minimum mean-square error (WMMSE) [Shi *et al.*, TSP 2011]
  - ⇒ Reformulate the optimization problem
  - ⇒ Implement block coordinate descent
  - ⇒ Leads to closed-form iteration formulas

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

- The optimal power p_i can be found as v_i^2
- WMMSE is an iterative approach to solve the optimization
  - ⇒ Update $\mathbf{u}$, $\mathbf{w}$, and $\mathbf{v}$ at each step by block coordinate descent
  - ⇒ Stop when change between consecutive steps is small enough

WMMSE: Update Equations

1. Initialize $v_i = p_{\max}$
2. **repeat** (for all i)
3. $\qquad w_i^{'} = w_i$
4. $\qquad u_i = \frac{H_{ii}v_i}{\sigma^2+\sum_j H_{ji}^2 v_j^2}$
5. $\qquad w_i = \frac{1}{1-u_i H_{ii} v_i}$
6. $\qquad v_i = \frac{\alpha_i u_i H_{ii} w_i}{\mu+\sum_j \alpha_j H_{ij}^2 u_j^2 w_j}$
7. **until** $\quad \sum_j \log w_j - \sum_j \log w_j^{'} < \epsilon$
8. $p_i = v_i^2$

WMMSE: Update Equations

1. Initialize $v_i = p_{\max}$
2. **repeat** (for all i)
3. $\quad w_i^{'} = w_i$
4. $\quad u_i = \frac{H_{ii} v_i}{\sigma^2 + \sum_j H_{ji}^2 v_j^2}$
5. $\quad w_i = \frac{1}{1 - u_i H_{ii} v_i}$
6. $\quad v_i = \frac{\alpha_i u_i H_{ii} w_i}{\mu + \sum_j \alpha_j H_{ij}^2 u_j^2 w_j}$
7. **until** $\quad \sum_j \log w_j - \sum_j \log w_j^{'} < \epsilon$
8. $p_i = v_i^2$

- 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 **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
  - $\Rightarrow$ CSI $\mathbf{H}$ as a weighted adjacency matrix of a directed graph
  - $\Rightarrow$ 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
  - $\Rightarrow$ CSI $\mathbf{H}$ as a weighted adjacency matrix of a directed graph
  - $\Rightarrow$ NSI $\mathbf{X}$ as a signal supported at the nodes
- $\mathbf{p}(\mathbf{H}, \mathbf{X}) = \Psi(\mathbf{H}, \mathbf{X}; \boldsymbol{\Theta})$, where Ψ is a K-layered GNN
  - $\Rightarrow$ $\boldsymbol{\Theta}$ 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
  - $\Rightarrow$ CSI $\mathbf{H}$ as a weighted adjacency matrix of a directed graph
  - $\Rightarrow$ NSI $\mathbf{X}$ as a signal supported at the nodes
- $\mathbf{p}(\mathbf{H}, \mathbf{X}) = \Psi(\mathbf{H}, \mathbf{X}; \mathbf{\Theta})$, where Ψ is a K-layered GNN
  - $\Rightarrow$ $\mathbf{\Theta}$ 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

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

REGNN

- Standard layered GNN architecture

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

- 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

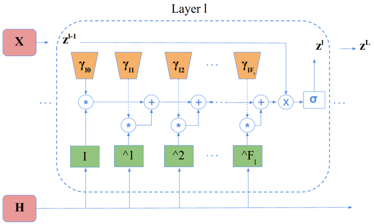
REGNN

- Standard layered GNN architecture

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

- 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

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

IGCNet

- Compute pairwise influence (interference) of each neighbor

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

- Local non-linear aggregation of neighborhood interference

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

IGCNet

- Compute pairwise influence (interference) of each neighbor

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

- Local non-linear aggregation of neighborhood interference

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

- Learn policy based on combination of channel with interference

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

IGCNet

- Compute pairwise influence (interference) of each neighbor

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

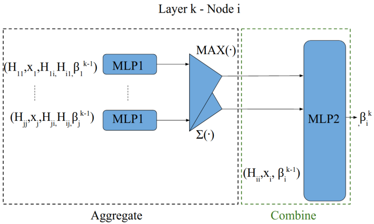
- Local non-linear aggregation of neighborhood interference

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

- Learn policy based on combination of channel with interference

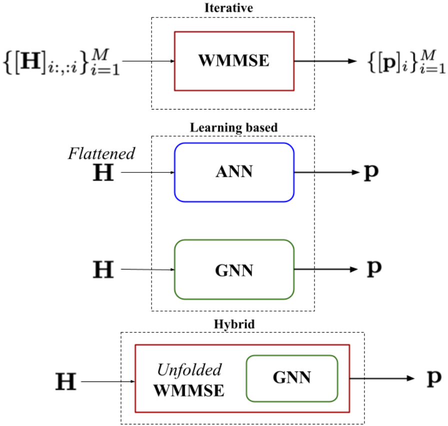
$$\beta_i^{k} = MLP2(\alpha_i^{k}, H_{ii}, \beta_i^{k-1}, x_i) \qquad \forall i$$

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
  $\Rightarrow$ Good performance but slow and/or expensive
- Each step computes variables of interest from a set of parameters
- Limit number of iterations $\Rightarrow$ suboptimal performance
- Need more efficient parameter update for faster convergence

Algorithm Unrolling

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

Algorithm Unrolling

- Iterative algorithms are long cascades of iterative steps

  $\Rightarrow$ Good performance but slow and/or expensive
- Each step computes variables of interest from a set of parameters
- Limit number of iterations $\Rightarrow$ suboptimal performance
- Need more efficient parameter update for faster convergence
- Algorithm Unrolling - learn from data Monga, Li & Eldar, 2019 arxiv, 2021 IEEE SPM)

  $\Rightarrow$ Supervised/Un-supervised gradient feedback
- Iterations $\Rightarrow$ layers, Parameters $\Rightarrow$ neural networks
- More interpretable operations, easy to follow update trajectory
- Once trained, can be used off-the-shelf $\Rightarrow$ Effective for online solutions

Proposed Method

- UWMMSE update rules at arbitrary layer k

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

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

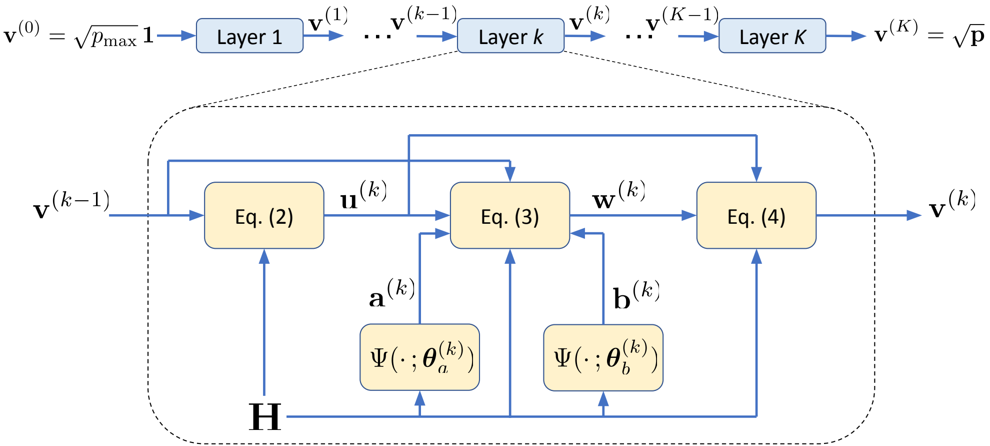
$$w_i^{(k)} = \frac{a_i^{(k)}}{1 - u_i^{(k)} h_{ii} v_i^{(k-1)}} + b_i^{(k)}, \qquad \forall i \tag{3}$$

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

- **v** and **u** as transmitter and receiver variables
- **w** as a tunable parameter
- $a = 1, b = 0$ yields classical solution

Block Diagram

- k^{th} layer of the model is shown below



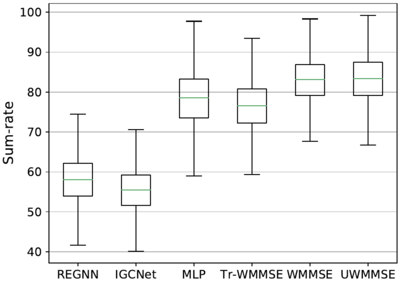
Simulation Results - 1

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

Performance Comparison

$\Rightarrow$ Network size $M = 20$;

$\Rightarrow$ $K = 4, K_{\max} = 100$



Time Comparison

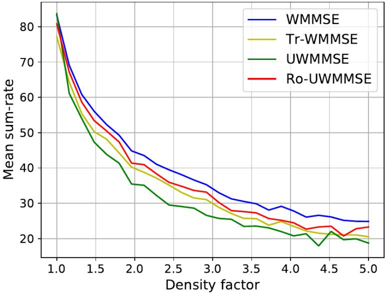
Algorithm	Training time (m)	Test sum-rate	Test time (ms)
WMMSE	-	82.94	16
Tr-WMMSE	-	76.49	1.0
MLP	0.5	78.17	3.2
REGNN	15	57.92	2.5
IGCNet	5	55.30	3
UWMMSE	15	**83.21**	**2.0**

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

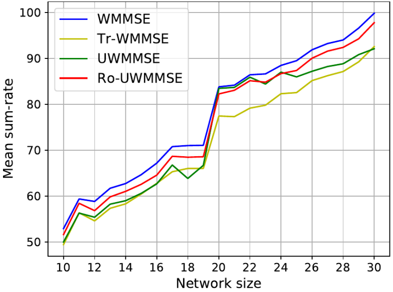
Simulation Results - 2

- Simulating dynamic network topologies
  - ⇒ Nodes in motion
  - ⇒ Insertion / Deletion of nodes

- **Variation in Spatial Density**



- **Variation in Network Size**



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

Optimal Power Allocation & Beamforming - MIMO Case

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 $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{M\times M\times R\times T}$

  $\Rightarrow$ Encodes channel characteristics

  $\Rightarrow$ $[\boldsymbol{\mathcal{H}}]_{ji::} = \mathbf{H}_{ji} \in \mathbb{R}^{R\times T}$ represents a MIMO channel from i to $r(j)$

  $\Rightarrow$ Channel between Tx-antenna k and Rx-antenna l is given by

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

  $\Rightarrow$ where $H^{P}_{jilk} \propto \mathrm{dist}(i, r(j))^{-k}$ for all l, k and $H^{F}_{jilk} \sim \mathrm{Rayleigh}(\alpha)$

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 $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{M\times M\times R\times T}$
  - $\Rightarrow$ Encodes channel characteristics
  - $\Rightarrow$ $[\boldsymbol{\mathcal{H}}]_{ji::} = \mathbf{H}_{ji} \in \mathbb{R}^{R\times T}$ represents a MIMO channel from i to $r(j)$
  - $\Rightarrow$ Channel between Tx-antenna k and Rx-antenna l is given by

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

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

Problem Description

Given the CSI tensor $\boldsymbol{\mathcal{H}}$, and a network utility function $u(\boldsymbol{\mathcal{H}}, \boldsymbol{\mathcal{V}}, \mathbf{p})$, determine the optimal power allocation $\mathbf{p}$ and $\boldsymbol{\mathcal{V}}$

Problem Description

Given the CSI tensor $\boldsymbol{\mathcal{H}}$, and a network utility function $u(\boldsymbol{\mathcal{H}}, \boldsymbol{\mathcal{V}}, \mathbf{p})$, determine the optimal power allocation $\mathbf{p}$ and $\boldsymbol{\mathcal{V}}$

- Power Constraint: Maximum power at each node $\Rightarrow p_i \leq P_{\max}$
- Network utility: sum rate across nodes

Problem Description

Given the CSI tensor $\boldsymbol{\mathcal{H}}$, and a network utility function $u(\boldsymbol{\mathcal{H}}, \boldsymbol{\mathcal{V}}, \mathbf{p})$, determine the optimal power allocation $\mathbf{p}$ and $\boldsymbol{\mathcal{V}}$

- Power Constraint: Maximum power at each node $\Rightarrow p_i \leq P_{\max}$
- Network utility: sum rate across nodes
- Data rate at receiver i is given by (for noise variance σ^2)

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

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

- Maximize weighted sum-rate $\sum_{i=1}^{M} \alpha_i c_i$
- Seeking a function $\Psi(\boldsymbol{\mathcal{H}})$

Classical Approach

- Weighted minimum mean-square error (WMMSE)
  - $\Rightarrow$ Reformulate the optimization problem [Shi *et al.*, TSP 2011]
  - $\Rightarrow$ Implement block coordinate descent
  - $\Rightarrow$ Leads to closed-form iteration formulae

$$\min_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}},\boldsymbol{\mathcal{V}}} \sum_{i=1}^{M} \left(\mathrm{Tr}(\mathbf{W}_i \mathbf{E}_i) - \log\det \mathbf{W}_i\right)$$

- $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{M\times R\times d}$ is the receiver beamformer tensor
- $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{M\times d\times d}$ is the node weight tensor

Classical Approach

- Weighted minimum mean-square error (WMMSE)
  - ⇒ Reformulate the optimization problem [Shi *et al.*, TSP 2011]
  - ⇒ Implement block coordinate descent
  - ⇒ Leads to closed-form iteration formulae

$$\min_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}},\boldsymbol{\mathcal{V}}} \sum_{i=1}^{M} \left(\mathrm{Tr}(\mathbf{W}_i \mathbf{E}_i) - \log\det \mathbf{W}_i\right)$$

- $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{M \times R \times d}$ is the receiver beamformer tensor
- $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{M \times d \times d}$ is the node weight tensor
- WMMSE is an iterative approach to solve the optimization
  - ⇒ Update $\boldsymbol{\mathcal{U}}$, $\boldsymbol{\mathcal{W}}$, and $\boldsymbol{\mathcal{V}}$ at each step by block coordinate descent
  - ⇒ Stop if change between consecutive steps is small enough

Graph-based and model-informed ML solution

- Use neural networks to learn the optimal power allocation $\mathbf{p}(\boldsymbol{\mathcal{H}})$
- Graph neural networks are good candidates to model this allocation
  - $\Rightarrow \mathbf{p}(\boldsymbol{\mathcal{H}}) = \Psi(\boldsymbol{\mathcal{H}}; \boldsymbol{\Theta})$, where Ψ is an K-layered GNN
  - $\Rightarrow \boldsymbol{\Theta}$ is the set of trainable weights

Graph-based and model-informed ML solution

- Use neural networks to learn the optimal power allocation $\mathbf{p}(\boldsymbol{\mathcal{H}})$
- Graph neural networks are good candidates to model this allocation
  - $\Rightarrow$ $\mathbf{p}(\boldsymbol{\mathcal{H}}) = \Psi(\boldsymbol{\mathcal{H}}; \boldsymbol{\Theta})$, where Ψ is an K-layered GNN
  - $\Rightarrow$ $\boldsymbol{\Theta}$ is the set of trainable weights
- Built-in scalability
  - $\Rightarrow$ We can train and test with different sizes of systems
- Exploit the right symmetries
  - $\Rightarrow$ Permutation equivariance is a natural feature for power allocation

Graph-based and model-informed ML solution

- Use neural networks to learn the optimal power allocation $\mathbf{p}(\boldsymbol{\mathcal{H}})$
- Graph neural networks are good candidates to model this allocation
  - $\Rightarrow$ $\mathbf{p}(\boldsymbol{\mathcal{H}}) = \Psi(\boldsymbol{\mathcal{H}}; \boldsymbol{\Theta})$, where Ψ is an K-layered GNN
  - $\Rightarrow$ $\boldsymbol{\Theta}$ is the set of trainable weights
- Built-in scalability
  - $\Rightarrow$ We can train and test with different sizes of systems
- Exploit the right symmetries
  - $\Rightarrow$ Permutation equivariance is a natural feature for power allocation
- Model-informed solution via algorithm unfolding
- Layers in a neural architecture inspired by iterations of WMMSE
  - $\Rightarrow$ More interpretable operations
  - $\Rightarrow$ Easy to fall back into classical solution

Proposed Method

- UWMMSE update rules at arbitrary layer k

$$\mathbf{a}^{(k)} = \Psi(\bar{\mathcal{H}}; \theta_a), \quad \mathbf{b}^{(k)} = \Psi(\bar{\mathcal{H}}; \theta_b), \tag{2}$$

$$\mathbf{U}_i^{(k)} = \left(\sum_{j \neq i} \mathbf{H}_{ij} \mathbf{V}_j^{(k-1)} \mathbf{V}_j^{(k-1)\top} \mathbf{H}_{ij}^\top + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{H}_{ii} \mathbf{V}_i^{(k-1)} \qquad \forall i \tag{3}$$

$$\mathbf{W}_i^{(k)} = [\mathbf{a}^{(k)}]_i \left(\mathbf{I} - \mathbf{U}_i^{(k)\top} \mathbf{H}_{ii} \mathbf{V}_j^{(k-1)} \right)^{-1} + [\mathbf{b}^{(k)}]_i \qquad \forall i \tag{4}$$

$$\mathbf{V}_i^{(k)} = \beta \left(\left(\sum_{j \neq i} \mathbf{H}_{ij}^\top \mathbf{U}_j^{(k)} \mathbf{W}_j^{(k)} \mathbf{U}_j^{(k)\top} \mathbf{H}_{ij} \right)^{-1} \mathbf{H}_{ii}^\top \mathbf{U}_i^{(k)} \mathbf{W}_i^{(k)} \right) \qquad \forall i \tag{5}$$

Proposed Method

- UWMMSE update rules at arbitrary layer k

$$\mathbf{a}^{(k)} = \Psi(\bar{\mathcal{H}}; \theta_a), \quad \mathbf{b}^{(k)} = \Psi(\bar{\mathcal{H}}; \theta_b), \tag{2}$$

$$\mathbf{U}_i^{(k)} = \left(\sum_{j \neq i} \mathbf{H}_{ij} \mathbf{V}_j^{(k-1)} \mathbf{V}_j^{(k-1)\top} \mathbf{H}_{ij}^\top + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{H}_{ii} \mathbf{V}_i^{(k-1)} \quad \forall i \tag{3}$$

$$\mathbf{W}_i^{(k)} = [\mathbf{a}^{(k)}]_i \left(\mathbf{I} - \mathbf{U}_i^{(k)\top} \mathbf{H}_{ii} \mathbf{V}_j^{(k-1)}\right)^{-1} + [\mathbf{b}^{(k)}]_i \quad \forall i \tag{4}$$

$$\mathbf{V}_i^{(k)} = \beta\left(\left(\sum_{j \neq i} \mathbf{H}_{ij}^\top \mathbf{U}_j^{(k)} \mathbf{W}_j^{(k)} \mathbf{U}_j^{(k)\top} \mathbf{H}_{ij}\right)^{-1} \mathbf{H}_{ii}^\top \mathbf{U}_i^{(k)} \mathbf{W}_i^{(k)}\right) \quad \forall i \tag{5}$$

- β is a clipper to enforce power constraint

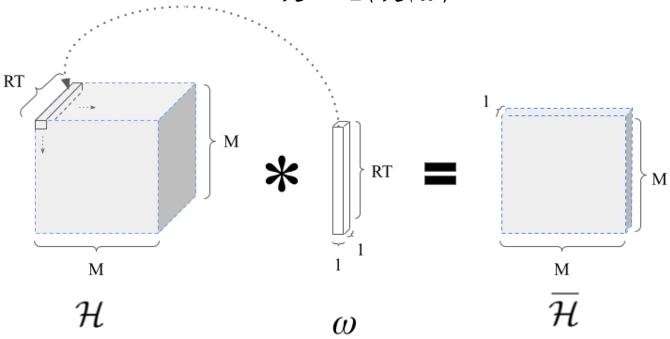
$$\beta(\mathbf{X}) = \begin{cases} \mathbf{X}, & \text{if } \mathrm{Tr}\left(\mathbf{X}\mathbf{X}^\top\right) \leq P_{\max}, \\ \mathbf{X} \cdot \frac{\sqrt{P_{\max}}}{||\mathbf{X}||_F}, & \text{otherwise}, \end{cases}$$

Tensor Reduction

- GNN Ψ requires CSI between i and $r(j)$ to be a scalar
- $\Phi(\boldsymbol{\mathcal{H}}; \omega) : \mathbb{R}^{M \times M \times R \times T} \to \mathbb{R}^{M \times M \times 1}$ where $\omega \in \mathbb{R}^{RT}$

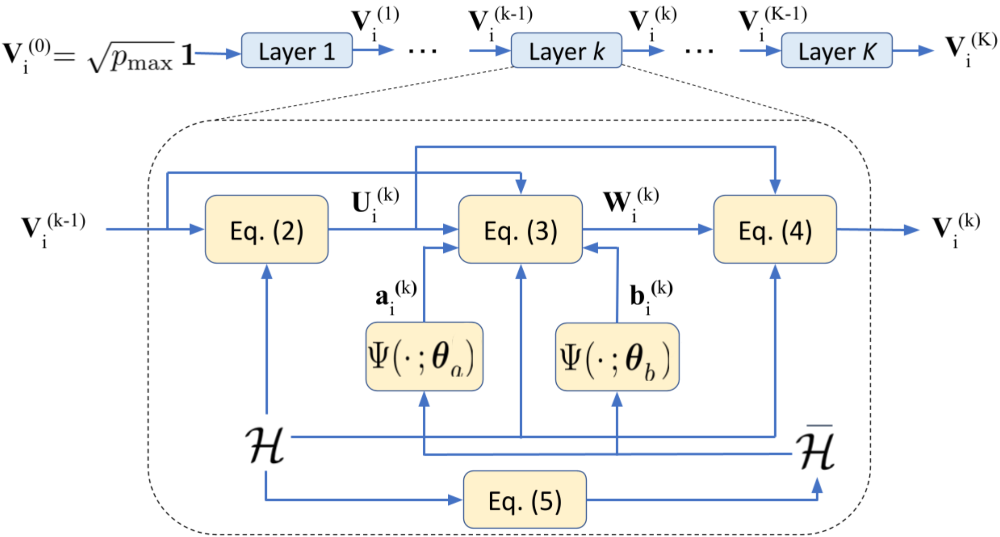
  $\Rightarrow$ Single-layered 1×1 depth-wise conv with shared weights
- Learnable weighted combination of RT coefficients at each node

$$\bar{\boldsymbol{\mathcal{H}}} = \Phi(\boldsymbol{\mathcal{H}}; \omega) \tag{6}$$



Block Diagram

- k^{th} layer of the model is shown below



Complexity Analysis

- Per-layer complexity of UWMMSE is $\mathcal{O}(M^2)$, same as WMMSE

Complexity Analysis

- Per-layer complexity of UWMMSE is $\mathcal{O}(M^2)$, same as WMMSE
- Linear layer Φ has $RT + 1$ parameters,

  $\Rightarrow$ Shared filter kernel allows for $\mathcal{O}(M^2)$ reduction

Complexity Analysis

- Per-layer complexity of UWMMSE is $\mathcal{O}(M^2)$, same as WMMSE
- Linear layer Φ has $RT+1$ parameters,

  $\Rightarrow$ Shared filter kernel allows for $\mathcal{O}(M^2)$ reduction
- Each of the two 2-layered GCNs Ψ, have $6h+2$ trainable weights θ

  $\Rightarrow$ h is the size of the hidden layer (typically ≤ 10)

Complexity Analysis

- Per-layer complexity of UWMMSE is $\mathcal{O}(M^2)$, same as WMMSE
- Linear layer Φ has $RT + 1$ parameters,

  $\Rightarrow$ Shared filter kernel allows for $\mathcal{O}(M^2)$ reduction
- Each of the two 2-layered GCNs Ψ, have $6h + 2$ trainable weights θ

  $\Rightarrow$ h is the size of the hidden layer (typically ≤ 10)
- Number of trainable weights is therefore $12h + RT + 5$

  $\Rightarrow$ Independent of the number of users M
- Very few trainable weights

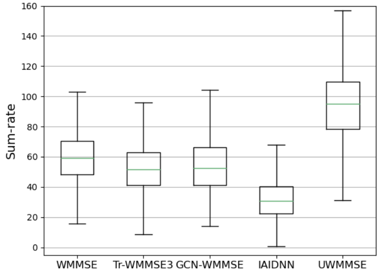
  $\Rightarrow$ Makes model easy to train

  $\Rightarrow$ Likely to generalize

Simulation Results - 1

- Random geometric graph with M node pairs
- Path loss & fading: Rayeligh, Rician, Network size $M = 20$

- **Performance Comparison**



- **Time Comparison**

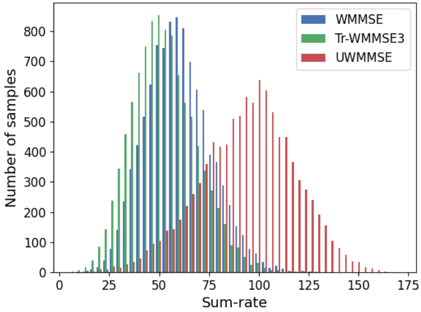
Algorithm	Training time (min)	Test time (sec)
WMMSE	-	1.305
Tr-WMMSE	-	0.047
IAIDNN	~ 10	0.64
GCN-WMMSE	~ 21	1.365
UWMMSE	~ 35	0.054

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

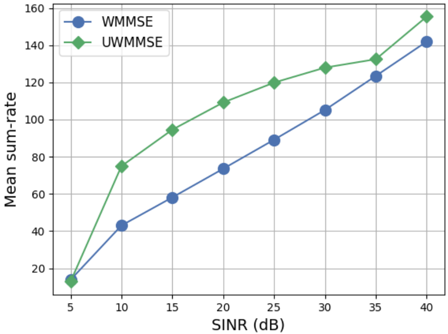
UWMMSE: Chowdhury *et al*, MILCOM'21, Asilomar'23, TWC'23

Simulation Results -2

- **Performance Comparison**



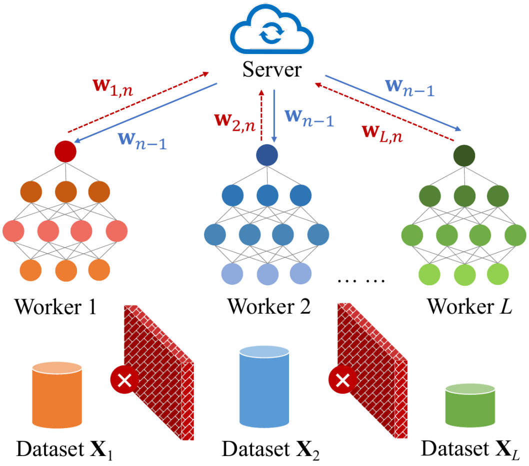
- **Generalization performance**
  - $\Rightarrow$ Over SINR



Chowdhury *et al*, MILCOM'21, TWC'23, Asilomar'23

Optimal Power Allocation - Federated Learning

Power allocation for wireless FL

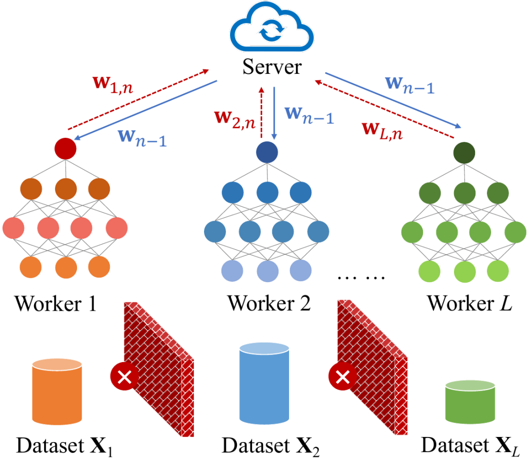


Critical step: upload local updates

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

Power allocation for wireless FL

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

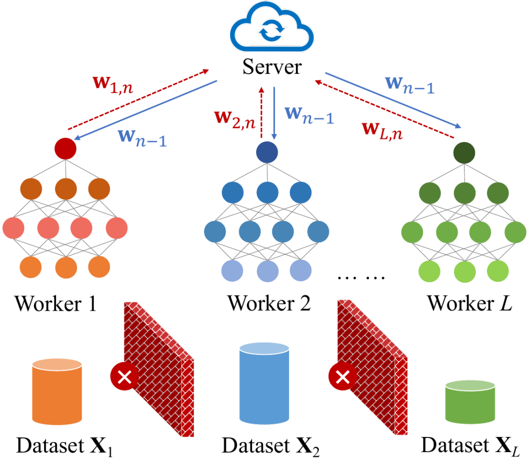


FL case is more challenging:

- Additional non-convex constraints on FL-specific requirements, e.g., delay and energy
- Ultimate goal of improving FL performance being 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

$\Rightarrow$ Primal-dual (PD) algorithm enhanced by graph learning

- Ultimate goal of improving FL performance being indirect to the communication objective

$\Rightarrow$ **Local data heterogeneity**

Problem formulation

Determine the power allocation policy $p^\star : \mathbb{R}^{L\times L} \to \mathbb{R}^L$ that solves the following optimization problem[1], subject to bounds on transmission rate, energy efficiency, and power

$$
\begin{aligned}
p^\star \quad &= \quad \underset{p}{\text{argmax}} \;\; g\left(\mathbb{E}_{\mathbf{H}\sim\mathcal{H}}\left[\text{PSR}(\mathbf{p},\mathbf{H})\right]\right), \\
\text{s.t.} \quad & r_{0,i} \le \mathbb{E}_{\mathbf{H}\sim\mathcal{H}}\left[R_i(\mathbf{p},\mathbf{H}) \,|\, p_i > 0\right], \\
& e_{0,i} \le \mathbb{E}_{\mathbf{H}\sim\mathcal{H}}\left[\frac{R_i(\mathbf{p},\mathbf{H})}{p_i + P_{\text{c},i}} \,\Big|\, p_i > 0\right], \forall i, \\
& \mathbf{p} = p(\mathbf{H}) \in [0, P_{\max}], \;\; \forall \mathbf{H},
\end{aligned}
$$

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

Problem formulation

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

$$p^\star = \underset{p}{\operatorname{argmax}} \; g\left(\mathbb{E}_{\mathbf{H}\sim\mathcal{H}}\left[\mathrm{PSR}(\mathbf{p}, \mathbf{H})\right]\right)$$

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

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

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

$$\begin{aligned}
\text{s.t.} \quad & \mathbf{y} \leq \mathbb{E}[\,\mathrm{PSR}(\mathbf{p}_\psi, \mathbf{H})\,], \\
& r_i \leq \mathbb{E}\left[R_i(\mathbf{p}_\psi, \mathbf{H}) \,|\, p_{\psi_i} > 0\right], \\
& e_i \leq \mathbb{E}\left[\frac{R_i(\mathbf{p}_\psi, \mathbf{H})}{p_{\psi_i} + P_{\mathrm{c}}} \,\middle|\, p_{\psi_i} > 0\right], \\
& r_i \in [r_{0,i}, +\infty), \\
& e_i \in [e_{0,i}, +\infty), \;\; \forall\, i, \\
& \mathbf{p}_\psi = p_\psi(\mathbf{H}) \in [0, P_{\max}], \;\; \forall\, \mathbf{H}.
\end{aligned}$$

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:

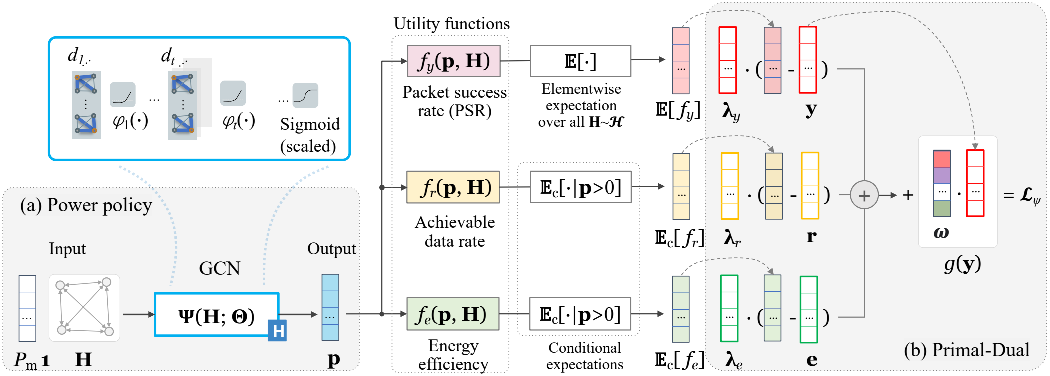
1. Learnable parameters $\boldsymbol{\Theta}$;
2. Primal variables $\mathbf{y}$, $\mathbf{r}$, and $\mathbf{e}$;
3. Dual variables $\boldsymbol{\lambda}_y$, $\boldsymbol{\lambda}_r$, and $\boldsymbol{\lambda}_e$.

Choosing GCN as Ψ constitutes our <u>p</u>rimal-<u>d</u>ual <u>g</u>raph convolutional (PDG) power network.

Proposed solution (before FL)

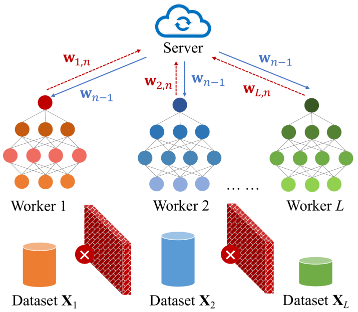
Two-stage solution with **two separate learning models**.

1. Before FL, train a power allocation policy model (see below).
2. During FL, apply the policy model to upload **local FL models** in each FL iteration that updates the **global FL model**.



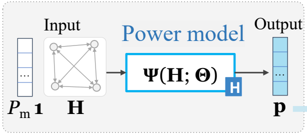
Proposed solution (during FL)

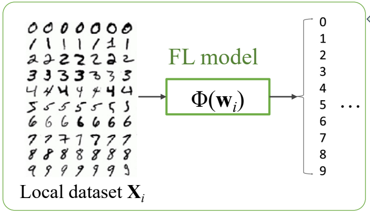
FL system:

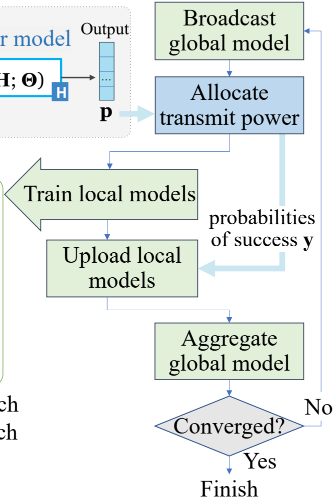


FL pipeline with power allocation policy:

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



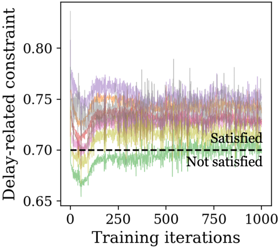




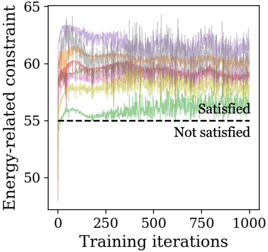
The local FL models are trained by each worker using their local datasets in each FL iteration.

PD learning curves for the power policy

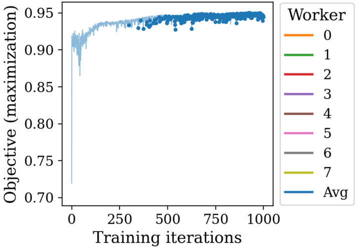
Learning curves[2] of PDG demonstrate convergence to
(a) delay constraint, (b) energy constraint, and (c) objective PSR.



(a)



(b)

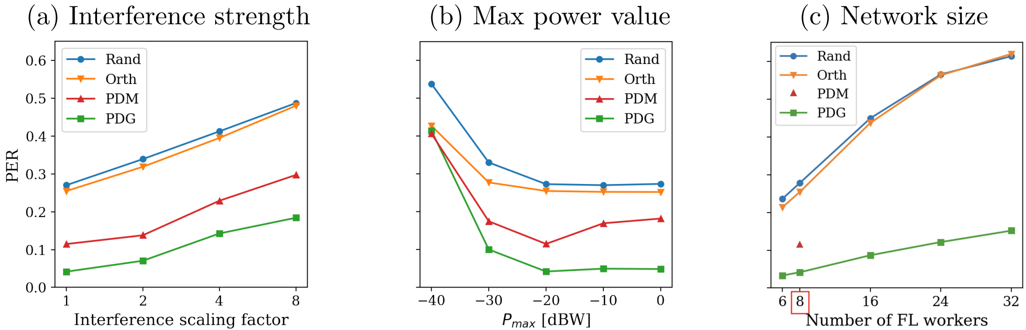


(c)

[2] Constraint 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:



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

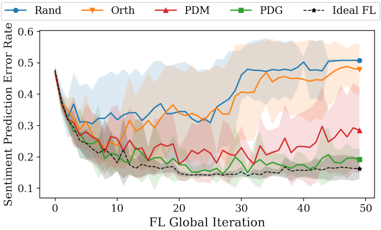
Orth: Chen *et al.*, TWC'20

FL performance on I.I.D. data

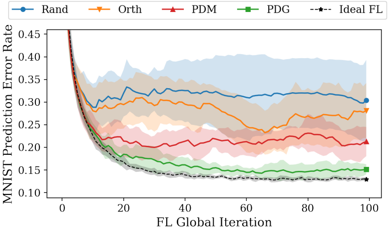
Tests on FL benckmark tasks: (a) NLP: IMDb sentiment classification, (b) MNIST digit classification, and (c) regresison: Air quality prediction.

Figures show global FL validation errors vs completed FL iterations.

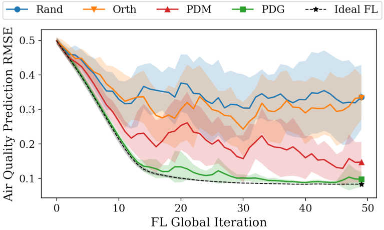
PDG consistently results in the best FL performance, close to ideal.



(a)



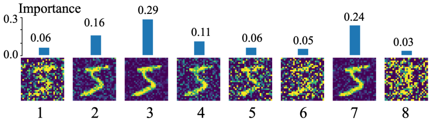
(b)



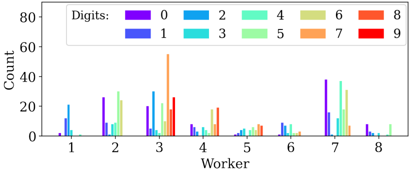
(c)

Non-I.I.D. case

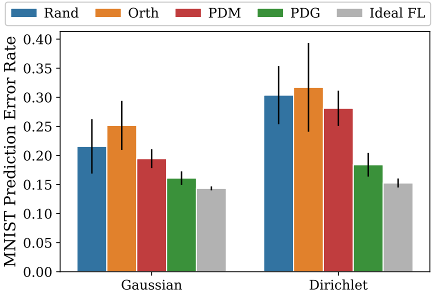
Local datasets may have different types, degrees, or patterns of random noise specific to the device or local environment.



(a) Heterogeneous AWGN. Bars denote corresponding worker weights that reflect local data quality.



(b) Skewed label histograms drawn from a Dirichlet distribution.

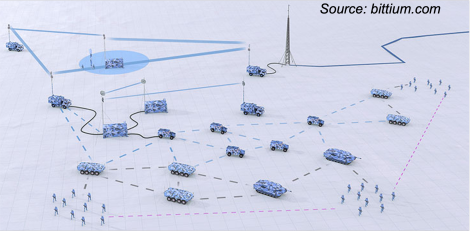


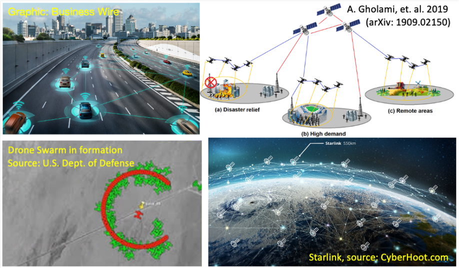
(c) Federated MNIST classification performance averaged across 5 random realizations. Results are shown for both Gaussian (noisy data) and Dirichlet (imbalanced labels) non-i.i.d. scenarios.

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

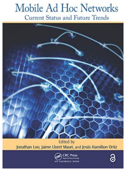
Part III: Graph-based ML for Wireless Networking

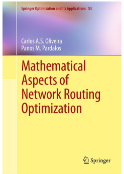
Introduction: wireless multihop networks

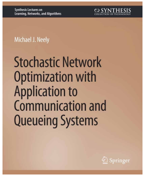




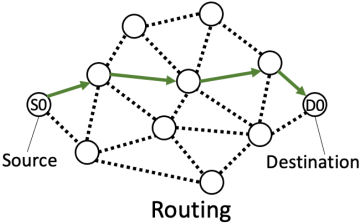
- User devices self-organization
- Infrastructure-less communications
  - ⇒ Military and disaster relief
- Emerging applications
  - ⇒ wireless backhaul, satellite constellation
  - ⇒ Traffic offloading (D2D, IoT)

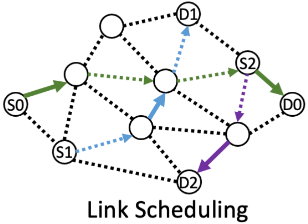






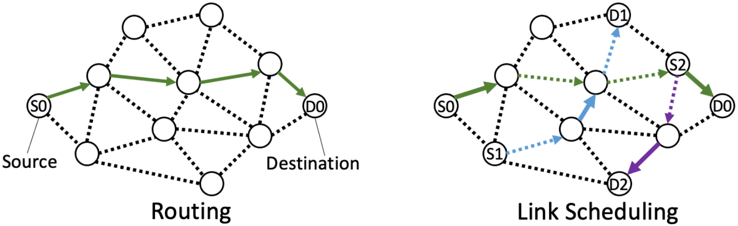
Fundamental wireless networking tasks





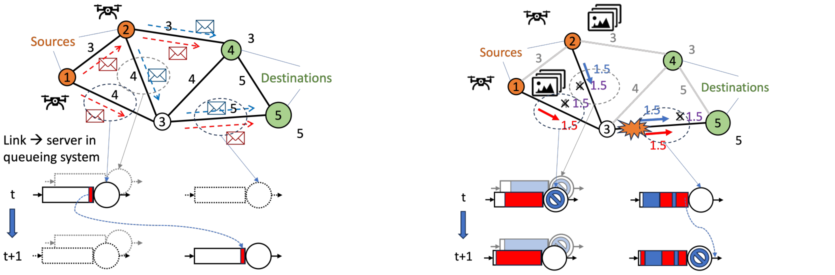
- Routing: send packets from source(s) to destination(s) through relay nodes
  - $\Rightarrow$ Path finding: 1-to-1 (unicast), 1-to-many: multi-cast, broadcast
  - $\Rightarrow$ Orchestration: cluster head election, virtual backbone establishment
- Link scheduling: decide which links to be activated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

Fundamental wireless networking tasks



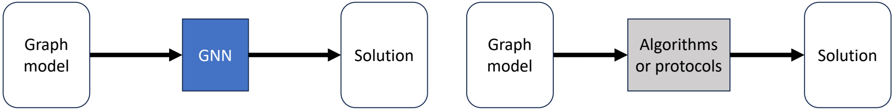
- Routing: send packets from source(s) to destination(s) through relay nodes
  - $\Rightarrow$ Path finding: 1-to-1 (unicast), 1-to-many: multi-cast, broadcast
  - $\Rightarrow$ Orchestration: cluster head election, virtual backbone establishment
- Link scheduling: decide which links to be activated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

Why wireless networking is so challenging?



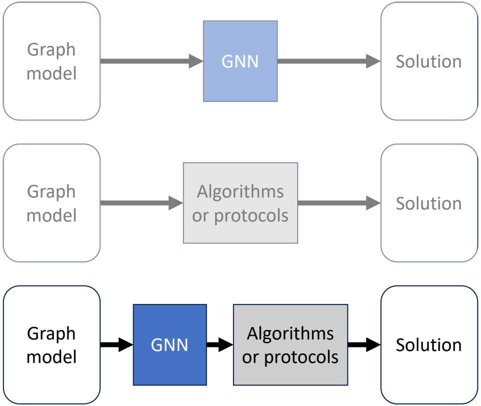
- Model: queueing networks subject to conflict constraints → no analytical model
- Instantaneous link rates fluctuate due to channel fading
- Changing network topology due to mobility
- Link capacity coupled with routing & scheduling decisions and input flow rates

GNNs for networking: opportunities and challenges

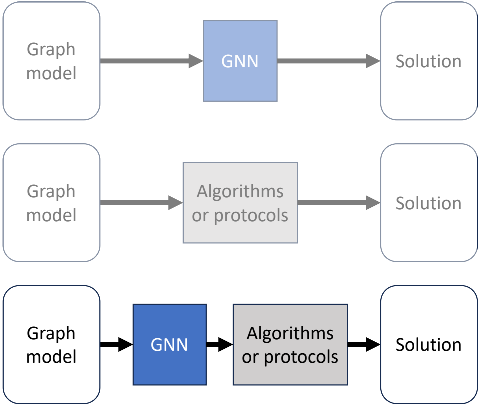


	GNNs	Algorithms
Transparency, interpretability explainability	poor	good
Theoretical bound & guarantees	poor	good
Natural aspect of networks	**opportunities**	limited
Representation of network structure	yes	limited
Learning from graph-structured data	yes	no
Inductive bias: permutation equivariance	yes	yes
Engineering aspect of networks	**challenges**	easy
Formulate networking tasks as link prediction, node classification, graph embedding	challenging	–
Domain knowledge: observe rules, constraints	hard	easy
Discrete decision-making	hard	easy
Data labeling for supervised learning	hard	–
Overall limitation	**functionality**	**optimality**

Part III Overview



Part III Overview



Graph-based machine learning for

- Max-Weight link scheduling
- Repetitive combinatorial optimization
- Conflict-aware packet routing
- Rapid network simulation
- Summary & future applications

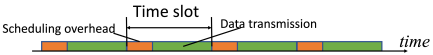
Let's start with a particular networking task

Link Scheduling with Graph Neural Networks [3][4]

[3] Z. Zhao, G. Verma, C. Rao, A. Swami and S. Segarra, "Distributed Scheduling Using Graph Neural Networks," IEEE ICASSP 2021, pp. 4720-4724

[4] 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

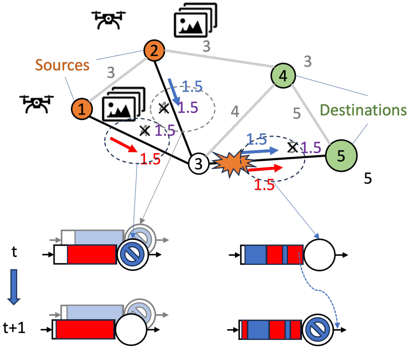
Link scheduling



Decide when and which links to be activated

Medium Access Control (MAC)

- Synchronized, time-slotted system
- Orthogonal multiple access
  - A resource block is exclusively assigned to an active link
  - In spatial, temporal, frequency, or code domains
- Bidirectional link $\rightarrow$ Undirected edge



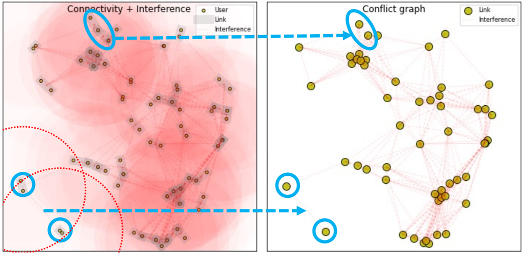
Link scheduling: graph modeling

Conflict graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- Vertex $v \in \mathcal{V} \rightarrow$ wireless link
- Edge $e \in \mathcal{E} \rightarrow$ conflict relationship between two wireless links that
  - $\Rightarrow$ share the same device (interface)
  - $\Rightarrow$ interfere each other if both activated

Vertex weights $\mathbf{u} \in \mathbb{R}_+^{|\mathcal{V}|} = [u(v)|v \in \mathcal{V}]$

- $u(v)$: utility of activating wireless link v



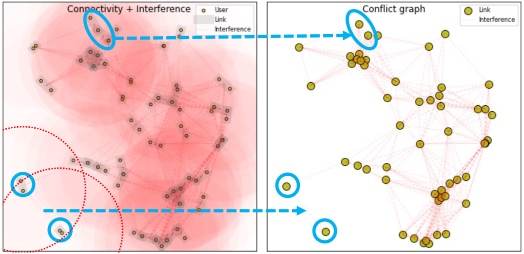
Link scheduling: graph modeling

Conflict graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- Vertex $v \in \mathcal{V} \rightarrow$ wireless link
- Edge $e \in \mathcal{E} \rightarrow$ conflict relationship between two wireless links that
  - $\Rightarrow$ share the same device (interface)
  - $\Rightarrow$ interfere each other if both activated

Vertex weights $\mathbf{u} \in \mathbb{R}_+^{|\mathcal{V}|} = [u(v)|v \in \mathcal{V}]$

- $u(v)$: utility of activating wireless link v



Utility function $u : \mathcal{V} \rightarrow \mathbb{R}_+$

- E.g., $u(v) = q(v)l(v)$, $u(v) = \min\{q(v), l(v)\}$ for throughput maximization
  - $\Rightarrow$ Queue length $q(v)$, link rate $l(v)$
  - $\Rightarrow$ Vector form $\mathbf{u} = \mathbf{q} \odot \mathbf{l}$, $\mathbf{u}_v = u(v), \mathbf{q}_v = q(v), \mathbf{l}_v = l(v)$

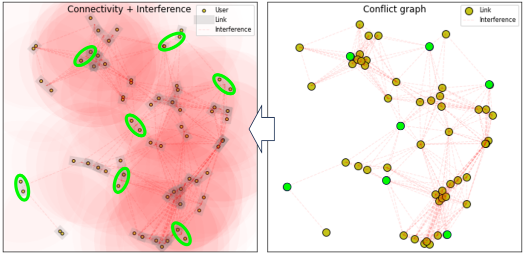
Max-Weight scheduling: MWIS formulation

Maximum weighted independent set (MWIS)

Consider a conflict graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ describe all the links and their conflict relationships in the wireless network, respectively, and a utility function $u : \mathcal{V} \to \mathbb{R}_+$. The optimal schedule is given by

$$\mathbf{v}^* = \underset{\mathbf{v} \subseteq \{0,1\}^{|\mathcal{V}|}}{\operatorname{argmax}} \; \mathbf{v}^\top \mathbf{u} \tag{7a}$$

$$\text{s.t. } \mathbf{v}_i + \mathbf{v}_j \leq 1 \;, \forall (i,j) \in \mathcal{E} \;. \tag{7b}$$



- MWIS problem is NP-hard
- Fast & distributed heuristics in practice

Distributed local greedy solver (LGS), $\mathcal{O}(\log|\mathcal{V}|)$

LGS[5] denoted as function $\hat{\mathbf{v}}_{Greedy} = h(\mathcal{G}, \mathbf{u})$, inspired by Ruby's algorithm[6]

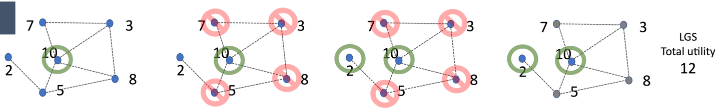
- All links initialized as undecided $\mathbf{v} = -\mathbf{1}$
- Link i is scheduled ($\mathbf{v}_i = 1$) if its utility exceeds all neighbors

$$u(i) > \max_{j \in \mathcal{N}(i)} u(j)$$

- Link i is muted ($\mathbf{v}_i = 0$) if one of its neighbors is scheduled
- Undecided nodes enter next iteration until all nodes are decided

An illustrative example

Local greedy solver (LGS)



---

[5] C. 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

[6] M. Luby. "A simple parallel algorithm for the maximal independent set problem." SIAM journal on computing 15.4 (1986): 1036-1053.

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)

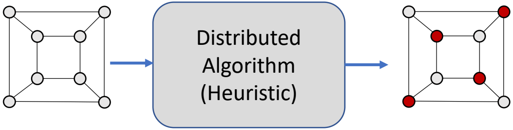
- Distributed & fast execution
- Generalize to different topologies
- Unable to encode relational constraints in COPs, e.g., $\mathbf{v}_i + \mathbf{v}_j \leq 1$, $\forall (v_i, v_j) \in \mathcal{E}$.

Graph convolutional layer (local form)

$$\mathbf{X}^{l}_{e*} = \sigma_l \left(\mathbf{X}^{l-1}_{e*} \boldsymbol{\Theta}^{l}_{0} + \left[\mathbf{X}^{l-1}_{e*} - \sum_{u \in \mathcal{N}_{\mathcal{G}^c}(e)} \frac{\mathbf{X}^{l-1}_{u*}}{\sqrt{d(e)d(u)}} \right] \boldsymbol{\Theta}^{l}_{1} \right)$$

Identical input → identical prediction

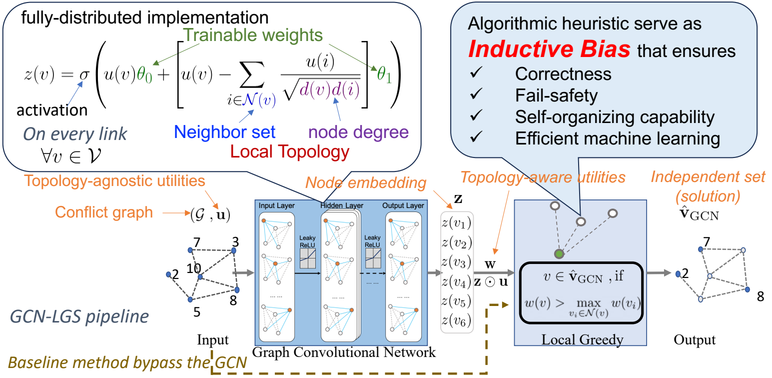
Constraint violations



Tie-breaking rules

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

GCN-enhanced local greedy solver (GCN-LGS)



Function notations:

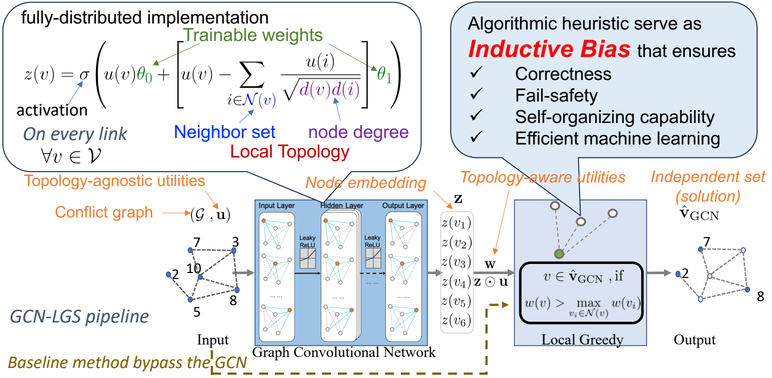
LGS: $\hat{\mathbf{v}}_{Greedy} = h(\mathcal{G}, \mathbf{u})$

GCN-LGS: $\hat{\mathbf{v}} = h(\mathcal{G}, \mathbf{z}\odot\mathbf{u})$

GCN: $\mathbf{z} = \Psi_{\mathcal{G}}(\mathbf{S}; \omega)$,

ω: trainable parameters

GCN-enhanced local greedy solver (GCN-LGS)



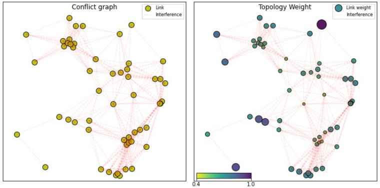
Function notations:

LGS: $\hat{\mathbf{v}}_{Greedy} = h(\mathcal{G}, \mathbf{u})$

GCN-LGS: $\hat{\mathbf{v}} = h(\mathcal{G}, \mathbf{z} \odot \mathbf{u})$

GCN: $\mathbf{z} = \Psi_{\mathcal{G}}(\mathbf{S}; \omega)$,

ω: trainable parameters



GCN-enhanced local greedy solver (GCN-LGS)

RICE DEVCOM ARMY RESEARCH LABORATORY

fully-distributed implementation

$$z(v) = \sigma\left(u(v)\theta_0 + \left[u(v) - \sum_{i\in\mathcal{N}(v)} \frac{u(i)}{\sqrt{d(v)d(i)}}\right]\theta_1\right)$$

Trainable weights

activation

On every link

$\forall v \in \mathcal{V}$

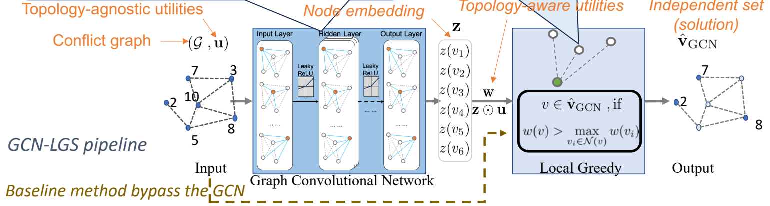
Neighbor set

node degree

Local Topology

Algorithmic heuristic serve as ***Inductive Bias*** that ensures

- ✓ Correctness
- ✓ Fail-safety
- ✓ Self-organizing capability
- ✓ Efficient machine learning



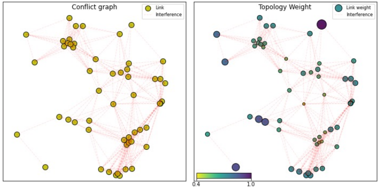
Function notations:

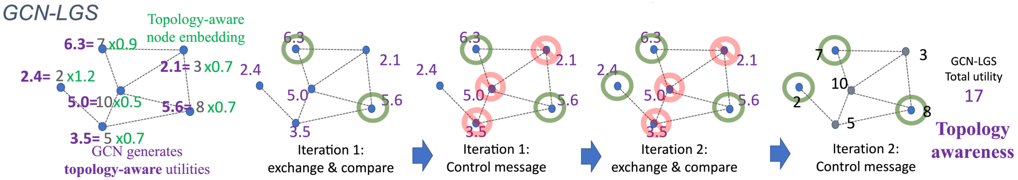
LGS: $\hat{\mathbf{v}}_{Greedy} = h(\mathcal{G}, \mathbf{u})$

GCN-LGS: $\hat{\mathbf{v}} = h(\mathcal{G}, \mathbf{z} \odot \mathbf{u})$

GCN: $\mathbf{z} = \Psi_{\mathcal{G}}(\mathbf{S}; \omega)$,

ω: trainable parameters





GCN Training: customized deterministic policy gradient[7]

Why reinforcement learning?

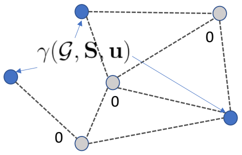
- Avoid data labeling – MWIS problem is NP-hard
- Treats non-differentiable LGS as part of the environment

[7] 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

GCN Training: customized deterministic policy gradient[7]

Why reinforcement learning?

- Avoid data labeling – MWIS problem is NP-hard
- Treats non-differentiable LGS as part of the environment



Gradient proxy

$$\omega^* = \underset{\omega}{\operatorname{argmax}} \; J(\omega) \tag{8a}$$

$$\text{s.t.} \;\; J(\omega) = \mathbb{E}_{(\mathcal{G},\mathbf{S},\mathbf{u})\sim\Omega}\left[\gamma(\mathcal{G},\mathbf{u},\mathbf{z})\right] \;, \tag{8b}$$

$$\gamma(\mathcal{G},\mathbf{u},\mathbf{z}) = \frac{\hat{\mathbf{v}}^\top \mathbf{u}}{\hat{\mathbf{v}}_{Greedy}^\top \mathbf{u}} \;, \tag{8c}$$

$$\hat{\mathbf{v}}_{Greedy} = h(\mathcal{G},\mathbf{u}) \;, \tag{8d}$$

$$\hat{\mathbf{v}} = h(\mathcal{G},\mathbf{z}\odot\mathbf{u}) \;, \tag{8e}$$

$$\mathbf{z} = \Psi_{\mathcal{G}}(\mathbf{S};\omega) \;. \tag{8f}$$

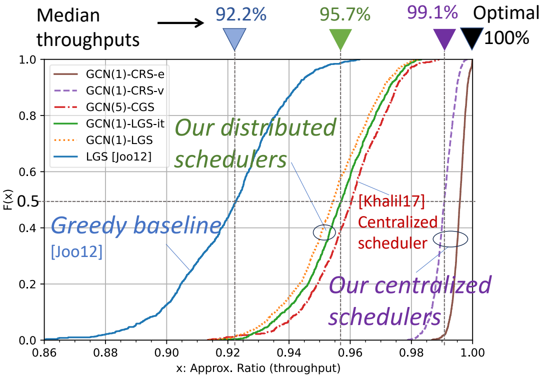
- Draw random network state $(\mathcal{G}, \mathbf{S}, \mathbf{u}) \sim \Omega$
- $\widehat{\nabla J(\omega)} = \gamma(\mathcal{G},\mathbf{u},\mathbf{z})\nabla\Psi_{\mathcal{G}}(\mathbf{S};\omega)\hat{\mathbf{v}}$, gradient
- Weight update $\omega = \omega + \alpha\widehat{\nabla J(\omega)}$, learning rate

[7] 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

Numerical results: throughput maximization[8]

Key Takeaways for GCN-LGS

- ▶ Close nearly half optimality gap
- ▶ Reusable GCN model
  - ⇒ Inner loop of LGS: GCN-LGS-it
  - ⇒ Centralized rollout search (CRS)
- ▶ Lightweight GCN: 2 trainable weights
- ▶ Failsafe: can fallback to vanilla LGS for basic functionality if GCN went crazy
- ▶ Low complexity: $\mathcal{O}(\log |\mathcal{V}|)$



- ▶ 100 nodes, 40 $\sim$ 60 links
- ▶ Utility function $u(v) = \min\left[r(v), q(v)\right]$
- ▶ Flooding traffic
- ▶ 100 graphs $\times$ 10 instances $\times$ 200 time steps

[8] 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

How to generalize GCN-LGS to broader networking tasks?

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

How to generalize GCN-LGS to broader networking tasks?

Graph-based deterministic policy gradient (GDPG-Twin) for repetitive combinatorial optimization problems (R-COPs)[9]

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

Combinatorial Optimization Problem (COP)

Typical formulation

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

COP

s.t. *Discrete constraint on nodes or edges*

Constraints defined on Graph, Hypergraph, or Simplicial Complex

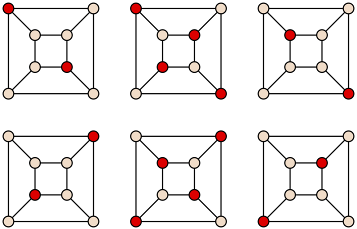
Characters

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

Maximum Weighted Independent Set

$$\mathbf{v}^* = \operatorname*{argmax}_{\mathbf{v} \subseteq \{0,1\}^{|\mathcal{V}|}} \ \mathbf{u}^\top \mathbf{v} \tag{9a}$$

$$\text{s.t. } \mathbf{v}_i + \mathbf{v}_j \leq 1 \ , \ \forall (i,j) \in \mathcal{E} \ . \tag{9b}$$



Source: Wikipedia – Maximal independent set

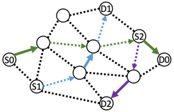
Repetitive Combinatorial Optimization Problem (R-COP)

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

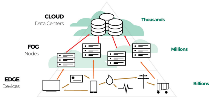
Routing & Scheduling in **communication networks**



Multi-object tracking in **computer vision**

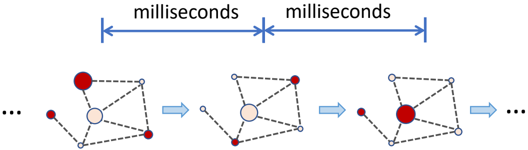
Vehicle routing problems in **distribution networks**

Resource allocation & job scheduling in **cloud, frog, edge computing**

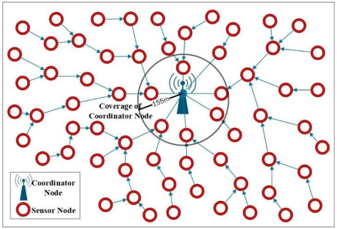


- Practical restrictions: limited runtime and/or distributed execution

Restrictions on runtime and distributed execution



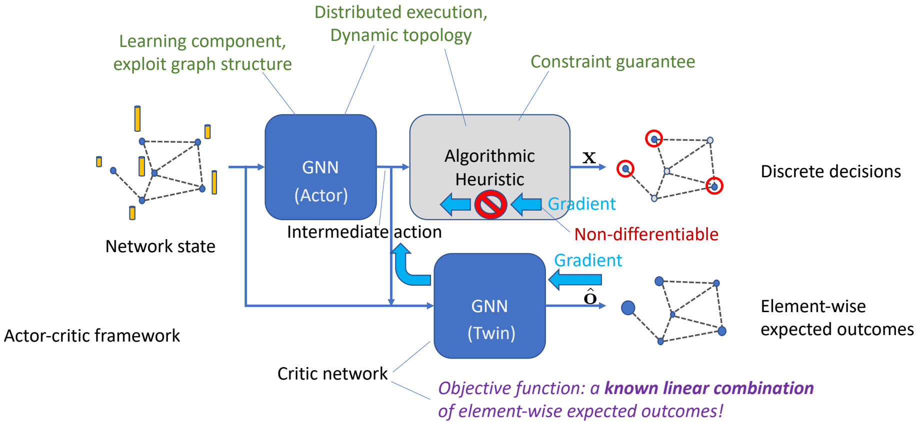
e.g., COP instances coming at data or video frame rates in wireless link scheduling or computer vision



Source: (D. Arı , M. Çıbuk and F. Ağgün , 2017)

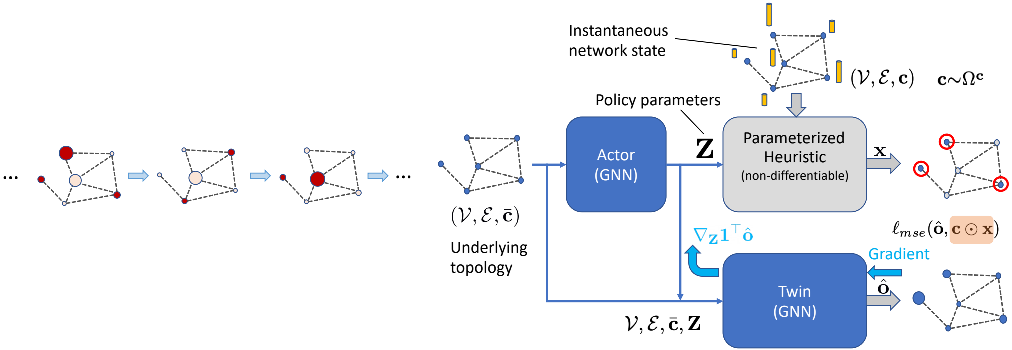
- Centralized COP solver
  - ⇒ High communication overhead to collect full network state to a server
  - ⇒ High computational complexity, scale up quickly by network size
  - ⇒ Single point of failure
- Distributed COP solver for scalability and robustness
  - ⇒ Fast & robust execution using only neighborhood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

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
  - $\Rightarrow$ 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

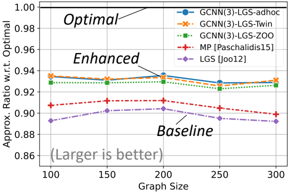


Figure 1: Approximation ratios (Larger is better) of the vanilla and GCNN-enhanced distributed heuristics for MWIS problem (max), w.r.t. the optimal solver.

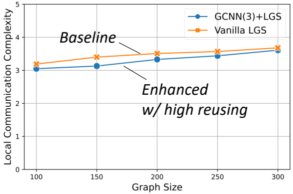


Figure 3: Average local communication complexity of GCNN-enhanced and vanilla LGS-MWIS solvers per instance, in rounds, excluding the GCNN ($N = \infty$).

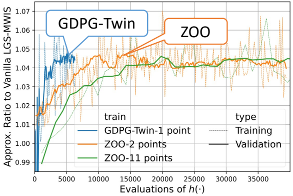


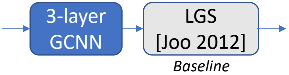
Figure 8: Performance trajectories of GCNN-enhanced LGS-MWIS trained by GDPG-Twin and ZOOs with 2-point and 11-point gradient estimations. Larger is better. GDPG-Twin needs fewer evaluations of $h(\cdot)$.

Approximation ratio

Execution local complexity

Training complexity

Benchmark: ZOO (zeroth-order optimization)

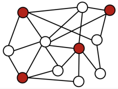


- Tested on 500 random graphs from **Erdős–Rényi** model
- Baseline: LGS[10], Benchmark: Zeroth-order optimization (ZOO)

[10] C. 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

Minimum Weighted Dominating Set



Source: J. Abernethy, CS 3510

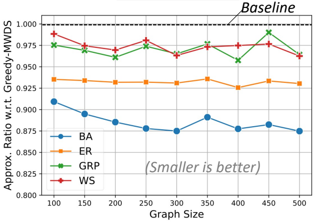
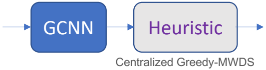
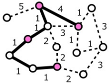


Figure 2: Approximation ratio (Smaller is better) of the GCNN-enhanced w.r.t. the vanilla Greedy-MWDS for MWDS problem (min) on 4 sets of random graphs.



Node Weighted Steiner Tree



Source: (M. Fujita, T. Kimura, & K. Jin'no, 2016)

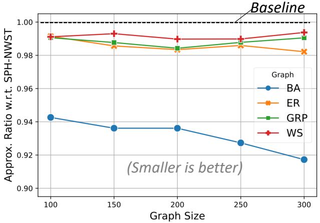
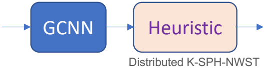
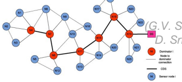


Figure 5: Approximation ratio (Smaller is better) of the GCNN-enhanced w.r.t. vanilla K-SPH-NWST for NWST problem on 4 sets of random graphs. NWST is a minimization (min) problem.



Minimum Weighted Connected Dominating Set



Source: (G.V. Shaamili Varsa, D. Sridharan, 2019)

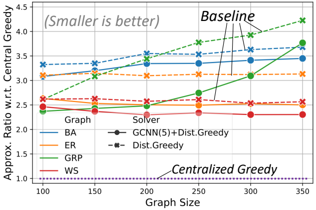
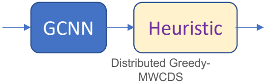
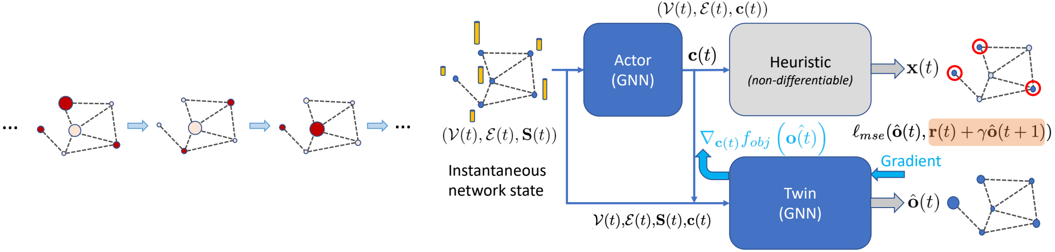


Figure 6: Approximation ratios (Smaller is better) of the vanilla and GCNN-enhanced distributed heuristics w.r.t. a centralized heuristic for MWCDS problem on 4 sets of random graphs. MWCDS is a min. problem.

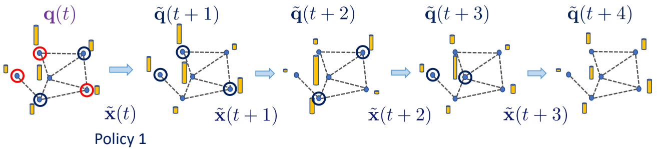


R-COP in graph-based Markov decision process



- Goal: optimize long-term system-level objective
  $\Rightarrow$ 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
  $\Rightarrow$ Consider future element-wise rewards
- Expected element-wise outcome $\hat{\mathbf{o}}(t) \approx \mathbf{o}(t) = \mathbb{E}\left[\mathbf{r}(t) + \gamma\hat{\mathbf{o}}(t+1)\right]$
- Gradient on intermediate action $\nabla_{\mathbf{c}(t)} f_{obj}\left(\hat{\mathbf{o}}(t)\right)$, f_{obj} is a linear function

Delay-oriented link scheduling



$$\omega^* = \operatorname*{argmax}_{\omega} \ \mathbb{E}_{i\in\mathcal{V},t\le T}\left[\mathbf{q}_i(t)\right] \tag{10a}$$

$$\text{s.t. } \mathbf{q}(t+1) = \mathbf{q}(t) + \mathbf{a}(t) - \mathbf{x}(t) \odot \min(\mathbf{l}(t), \mathbf{q}(t)) \ , \tag{10b}$$

$$\mathbf{x}(t) = h(\mathcal{V}, \mathcal{E}, \mathbf{q}(t), \mathbf{l}(t), \mathbf{a}(t); \omega) \ . \tag{10c}$$

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

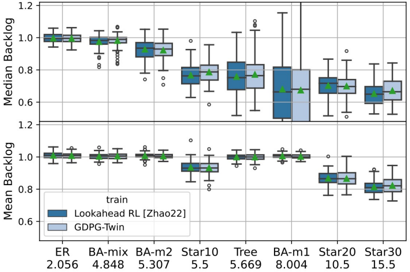


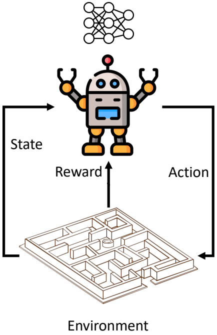
Figure 7: GDPG-Twin achieves similar network-wide mean and medium backlogs (smaller is better) of lookahead RL (Zhao et al., 2022b) in training a distributed link scheduler, using only 1/5 evaluations of $h(\cdot)$ of it.

- 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

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

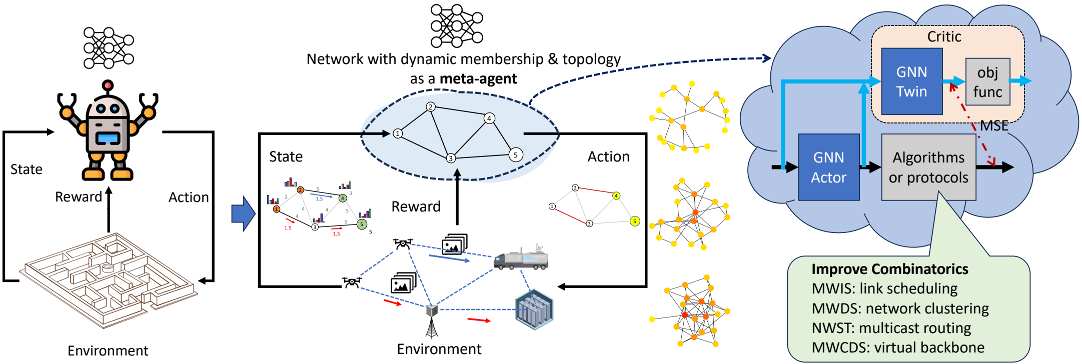
Recap on GDPG-Twin for R-COPs

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

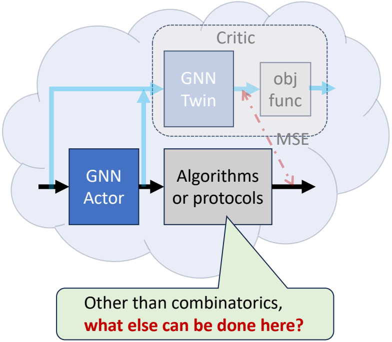


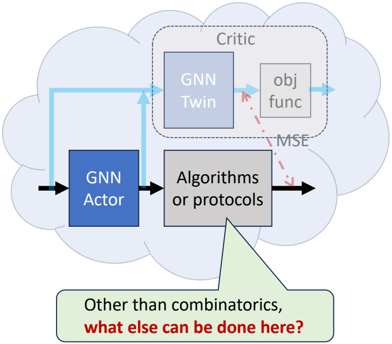
Recap on GDPG-Twin for R-COPs

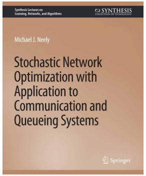
- Single-agent reinforcement learning for: scalar action & reward, state in regular domain



- GDPG-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


Critic
GNN
Twin
obj
func
MSE
GNN
Actor
Algorithms
or protocols
Other than combinatorics,
what else can be done here?





Encode Network Context into Backpressure Routing[a,b,c]

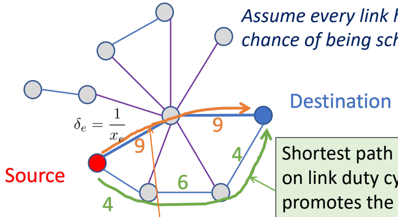
---

[a] Z. Zhao, B. Radojicic, G. Verma, A. Swami and S. Segarra, "Delay-Aware Backpressure Routing Using Graph Neural Networks," IEEE ICASSP 2023, pp. 1-5

[b] Z. Zhao, G. Verma, A. Swami and S. Segarra, "Enhanced Backpressure Routing Using Wireless Link Features," IEEE CAMSAP, 2023, pp. 271-275

[c] 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 Comms. and Netw.

From hop distance to conflict-aware shortest path



Assume every link has an equal chance of being scheduled

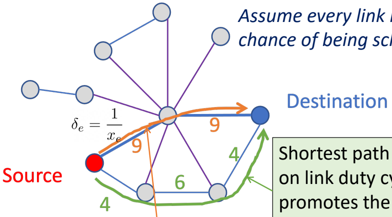
Shortest path bias based on link duty cycle promotes the **green route**, of which links have fewer conflicting neighbors, thus more likely to be scheduled (higher link duty cycle)

Shortest hop count bias promotes the **orange route**, of which both links have 8 conflicting neighbors, thus less likely being scheduled (link duty cycle = 1/9)

Insight: In wireless networks, links should not be treated equally since they introduce different latencies depending on local conflict topology.

From hop distance to conflict-aware shortest path

Assume every link has an equal chance of being scheduled



Shortest path bias based on link duty cycle promotes the **green route**, of which links have fewer conflicting neighbors, thus more likely to be scheduled (higher link duty cycle)

Shortest hop count bias promotes the **orange route**, of which both links have 8 conflicting neighbors, thus less likely being scheduled (link duty cycle = 1/9)

Insight: In wireless networks, links should not be treated equally since they introduce different latencies depending on local conflict topology.

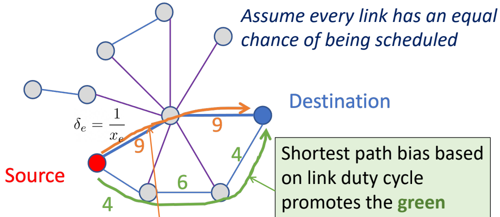
Graph modeling

- Connectivity graph $\mathcal{G}^n = (\mathcal{V}, \mathcal{E})$
- Conflict graph $\mathcal{G}^c = (\mathcal{E}, \mathcal{C})$

Per-hop distance $\delta_e, e \in \mathcal{E}$

- Shortest hop distance, $\delta_e = 1$
- With link duty cycle $0 < x_e < 1$
  - $\Rightarrow$ Definition 1: $\delta_e = 1/x_e$
  - $\Rightarrow$ Definition 2: $\delta_e = \frac{\bar{l}}{x_e l_e}$
- Link duty cycle predicted by GCNN $\mathbf{x} = \Psi_{\mathcal{G}^c}(\mathbf{S}; \omega)$

From hop distance to conflict-aware shortest path



Shortest hop count bias promotes the **orange route**, of which both links have 8 conflicting neighbors, thus less likely being scheduled (link duty cycle = 1/9)

Shortest path bias based on link duty cycle promotes the **green route**, of which links have fewer conflicting neighbors, thus more likely to be scheduled (higher link duty cycle)

Insight: In wireless networks, links should not be treated equally since they introduce different latencies depending on local conflict topology.

Graph modeling

- Connectivity graph $\mathcal{G}^n = (\mathcal{V}, \mathcal{E})$
- Conflict graph $\mathcal{G}^c = (\mathcal{E}, \mathcal{C})$

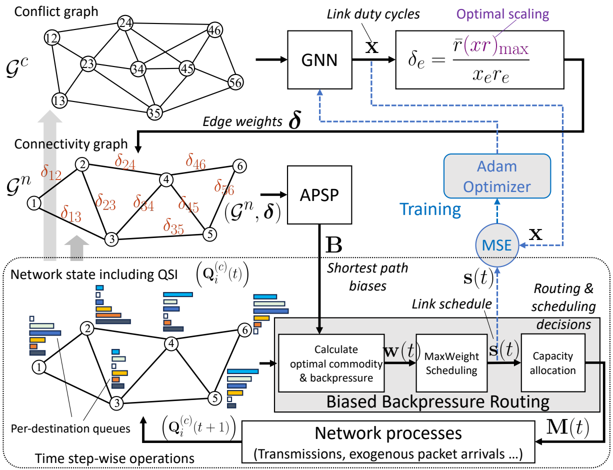
Per-hop distance $\delta_e, e \in \mathcal{E}$

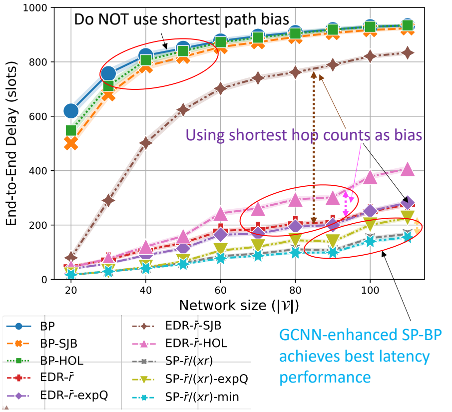
- Shortest hop distance, $\delta_e = 1$
- With link duty cycle $0 < x_e < 1$
  - $\Rightarrow$ Definition 1: $\delta_e = 1/x_e$
  - $\Rightarrow$ Definition 2: $\delta_e = \frac{\bar{l}}{x_e l_e}$
- Link duty cycle predicted by GCNN $\mathbf{x} = \Psi_{\mathcal{G}^c}(\mathbf{S}; \omega)$

Graph convolutional layer (local form)

$$\mathbf{X}^l_{e*} = \sigma_l \left(\mathbf{X}^{l-1}_{e*} \mathbf{\Theta}^l_0 + \left[\mathbf{X}^{l-1}_{e*} - \sum_{u \in \mathcal{N}_{\mathcal{G}^c}(e)} \frac{\mathbf{X}^{l-1}_{u*}}{\sqrt{d(e)d(u)}} \right] \mathbf{\Theta}^l_1 \right)$$

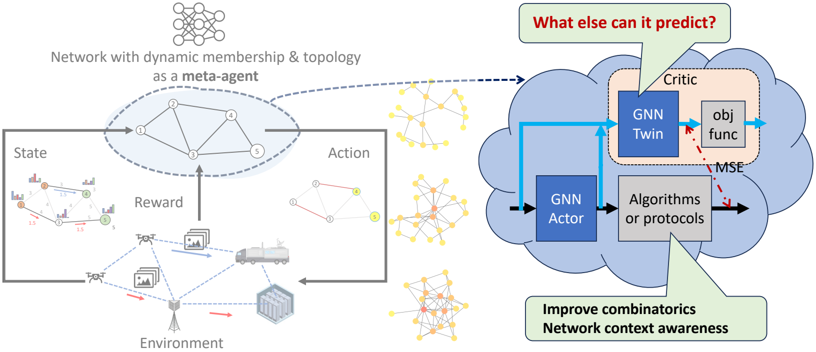
Conflict-aware shortest path for Backpressure routing[11]





SJB: Sojourn time of all packets in the queue [L. Hai, TVT, 2018]
HOL: Sojourn time of head-of-line packet in the queue [B. Ji, ToN, 2012]
EDR: Enhanced Dynamic Routing [M. Neely, JSAC, 2005]

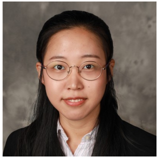
[11] Z. Zhao, B. Radojičić, G. Verma, A. Swami, S. Segarra, Biased Backpressure Routing Using Link Features and Graph Neural Networks, submitted to IEEE Trans. on Machine Learning In Comms. and Netw., under review.



Network with dynamic membership & topology
as a **meta-agent**
State
Reward
Action
Environment
What else can it predict?
Critic
GNN Twin
obj func
MSE
GNN Actor
Algorithms or protocols
Improve combinatorics
Network context awareness

New approach to network simulation & optimization

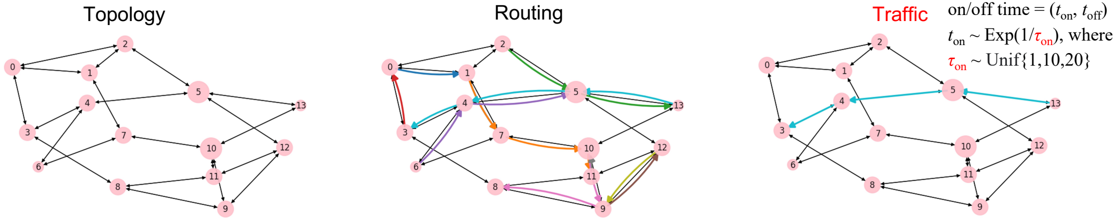
Network Digital Twin for Fast KPI prediction[12]



Credit: Boning Li

[12] 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.

Introduction of network simulators

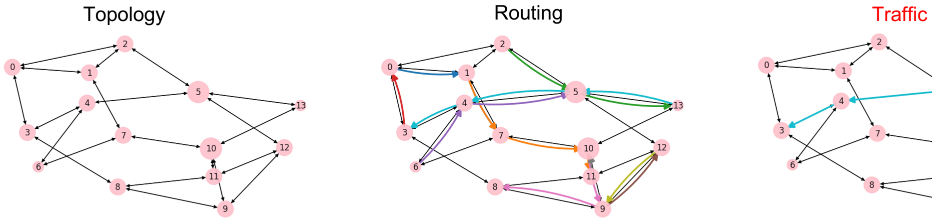


Example inputs: NSFNet (14 nodes, 42 links, 10 flows/paths)

- Each flow corresponds to a set of KPIs (key performance indicators)

  $\Rightarrow$ Guide the design, evaluation, and optimization of networks & protocols

Introduction of network simulators

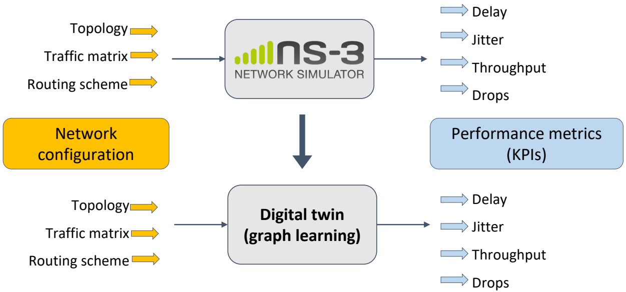


on/off time = $(t_{\text{on}}, t_{\text{off}})$
$t_{\text{on}} \sim \text{Exp}(1/\tau_{\text{on}})$, where
$\tau_{\text{on}} \sim \text{Unif}\{1,10,20\}$

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

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

What network digital twin can do?



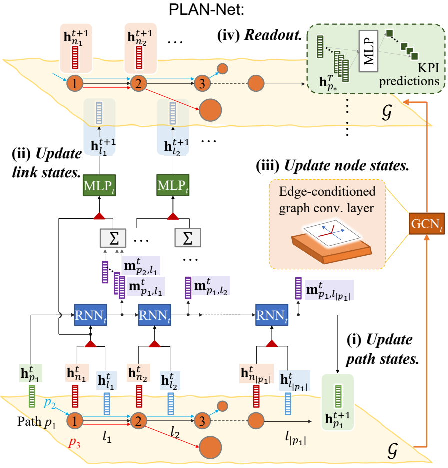
- Fast KPI prediction and differentiable process
- Digital twin of network simulators

  $\Rightarrow$ Predict KPIs rapidly (fast execution)

  $\Rightarrow$ Enable iterative optimization (fast execution)

  $\Rightarrow$ Training machine learning-based network solutions (differentiability)

Message-passing architecture of GNNs (PLAN-Net)



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

$\mathbf{h}_p^0 \leftarrow [\rho_p, 0, \cdots, 0]^\top, \forall p \in \mathcal{P}$
$\mathbf{h}_l^0 \leftarrow [c_l, 0, \cdots, 0]^\top, \forall l \in \mathcal{L}$
$\mathbf{h}_n^0 \leftarrow [d_n, 0, \cdots, 0]^\top, \forall n \in \mathcal{N}$
for $t = 0, 1, \cdots, T-1$ **do**

RNN
- *(i) Update path states.*
- **for** every path p in $\mathcal{P}$ **do**
  - **for** every link l in p **do**
    - $\mathbf{h}_p^t \leftarrow \mathrm{RNN}_t(\mathbf{h}_p^t, \mathrm{cat}[\mathbf{h}_l^t, \mathbf{h}_n^t])$, where $n = \mathrm{src}(l)$
      $\triangleright$ n is the source node of l
    - $\mathbf{m}_{p,l}^t \leftarrow \mathbf{h}_p^t$
  - **end for**
  - $\mathbf{h}_p^{t+1} \leftarrow \mathbf{h}_p^t$
- **end for**

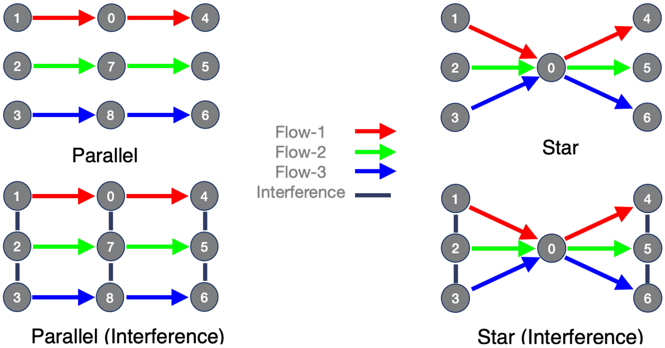
MLP
- *(ii) Update link states.*
- **for** every link l in $\mathcal{L}$ **do**
  - $\mathbf{h}_l^{t+1} \leftarrow \mathrm{MLP}_t(\mathrm{cat}[\mathbf{h}_l^t, \mathbf{h}_n^t, \mathrm{agg}\{\mathbf{m}_{p,l}^t \mid l \in p\}])$
    $\triangleright$ p is all paths that contain l
- **end for**

GCN
- *(iii) Update node states.*
- **for** every node n in $\mathcal{G}$ **do**
  - $\mathbf{h}_n^{t+1} \leftarrow \mathrm{GCN}_t(\mathrm{cat}[\mathbf{h}_n^t, \mathrm{agg}\{\mathbf{h}_l^t \mid l = L^+(n)\}]; \mathcal{G})$
    $\triangleright$ l is all links out of n
- **end for**

end for
(iv) *Readout.*
$\mathbf{y} = \mathrm{MLP}(\mathbf{h}_p^T)$

PLAN-Net (Path, Link, And Node)

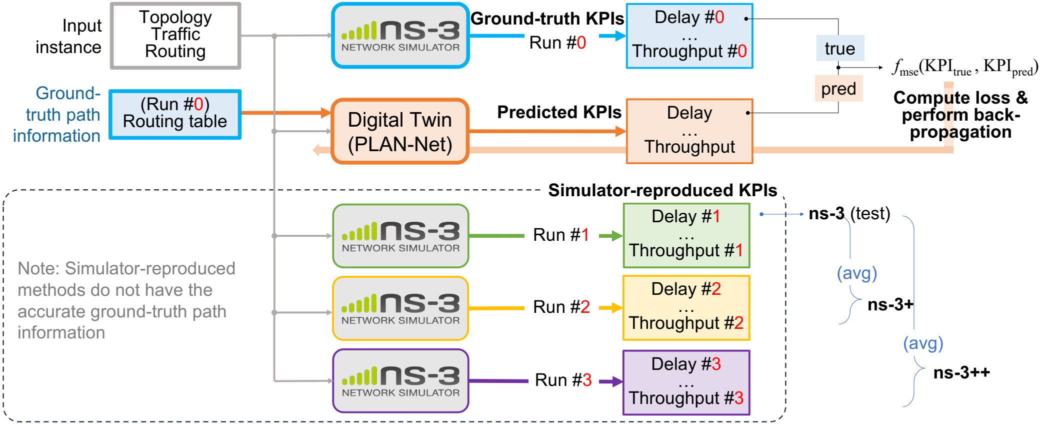
- PLAN-Net improves existing RouteNet[13] for wired networks
- Leverage node embeddings to distinguish different interference topologies



[13] K. 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.

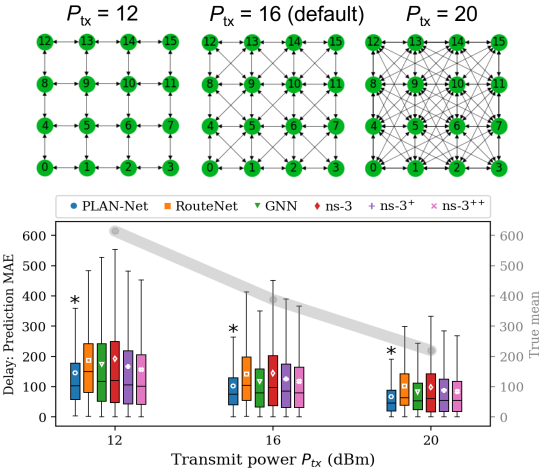
Training and evaluation

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



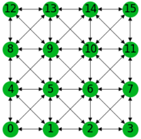
Numerical results: wireless networks of grid topology

- Alter transmit power to test for different levels of interference
- PLAN-Net achieves the lowest MAE
- Generalize to different network topologies

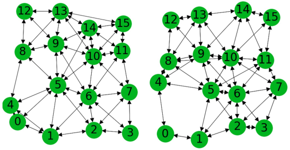


Numerical results: regular vs perturbed grid topologies

Regular grid



Perturbed grids



. . . . . .

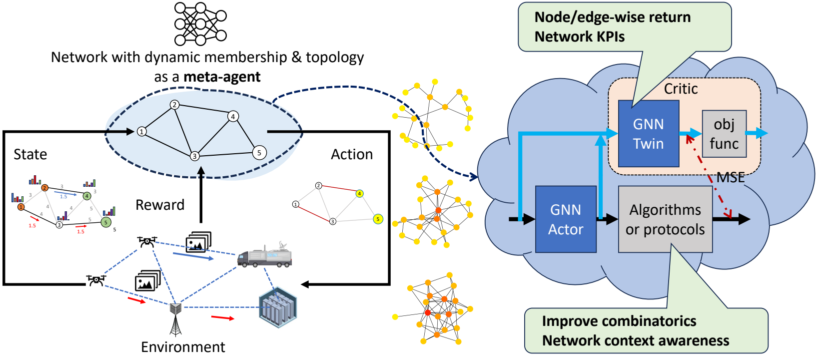
Method \ Metric	Delay Mean	Delay SD	Jitter Mean	Jitter SD	Throughput Mean	Throughput SD	Drops Mean	Drops SD
PLAN-Net	**0.23**	0.28	**0.35**	0.50	**0.19**	0.18	**0.24**	0.27
RouteNet	0.31	0.32	0.42	0.55	0.23	0.22	0.32	0.35
GNN	0.25	0.32	0.41	0.54	0.27	0.22	0.26	0.28
ns-3	0.32	0.39	0.46	0.88	0.25	0.26	0.33	0.41
ns-3^{+}	0.27	0.34	0.40	0.71	0.22	0.22	0.28	0.34
ns-3^{++}	0.26	0.32	0.38	0.65	0.20	0.21	0.27	0.32

Method \ Metric	Delay Mean	Delay SD	Jitter Mean	Jitter SD	Throughput Mean	Throughput SD	Drops Mean	Drops SD
PLAN-Net	0.29	0.31	0.46	0.71	0.23	0.21	0.29	0.31
RouteNet	0.34	0.34	0.48	0.74	0.25	0.24	0.33	0.37
GNN	0.32	0.35	0.54	0.77	0.29	0.24	0.30	0.32
ns-3	0.33	0.44	0.51	1.09	0.25	0.27	0.30	0.38
ns-3^{+}	0.29	0.37	0.45	0.86	0.22	0.23	0.26	0.32
ns-3^{++}	**0.27**	0.35	**0.43**	0.80	**0.21**	0.22	**0.25**	0.30

Key take-aways

- PLAN-Net is more accurate than a single-run of **ns-3**
- PLAN-Net can generalize to random perturbation of network topology
- PLAN-Net runs 1000x faster than **ns-3**, e.g., 100s → 0.01-0.1 s

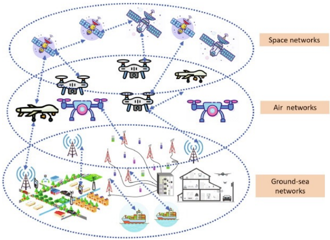
Summary & future work



- Graph-based ML: permutation invariant, distributed (scalable), no data labeling
  - $\Rightarrow$ Networks are dynamic and parallel & networking tasks are often discrete
- Hybrid ML pipelines $\rightarrow$ graph learning + domain knowledge = performance boost
- Digital twin $\rightarrow$ fast KPI prediction and action critic = new optimization tools

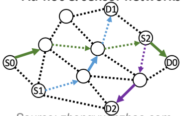
Potential applications of graph-based ML in large-scale networked systems

5G/6G networks



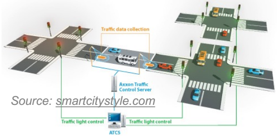
[SAA Hakeem, HH Hussein, HW Kim, 2022]

Ad-hoc & sensor networks



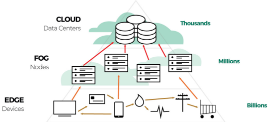
Source: zhongyuanzhao.com

Smart traffic light control system



Source: smartcitystyle.com

Cloud, fog, edge computing



Source: e-ziggurat.com

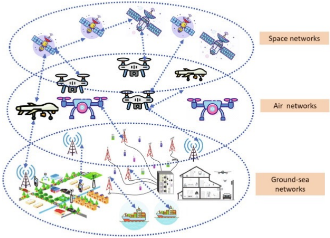
Operations Research

Source: caliper.com

[14] 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.

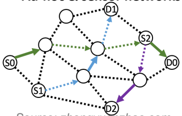
Potential applications of graph-based ML in large-scale networked systems

5G/6G networks



[SAA Hakeem, HH Hussein, HW Kim, 2022]

Ad-hoc & sensor networks

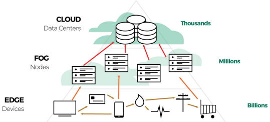


Source: zhongyuanzhao.com

Smart traffic light control system

Source: smartcitystyle.com

Cloud, fog, edge computing



Source: e-ziggurat.com

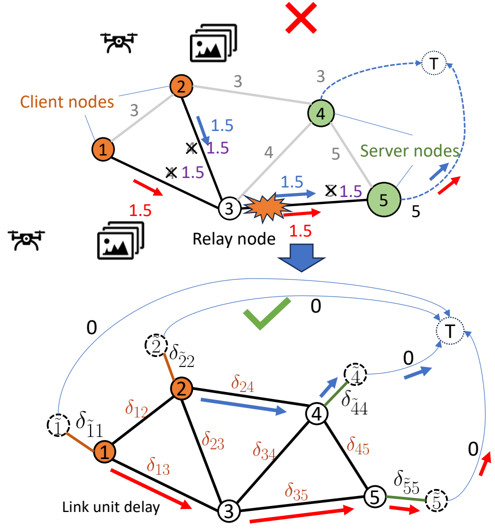
Operations Research

Source: caliper.com

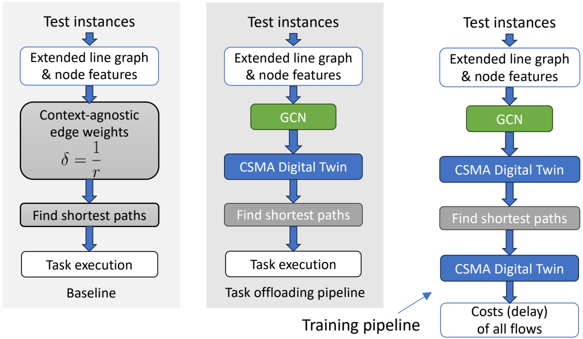
Edge computing & AI in wireless multihop networks[14]

[14] 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.

Congestion-aware distributed task offloading

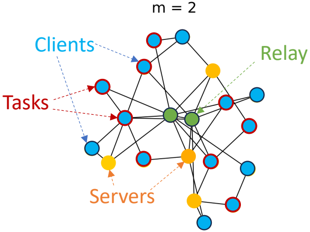


Extended graph

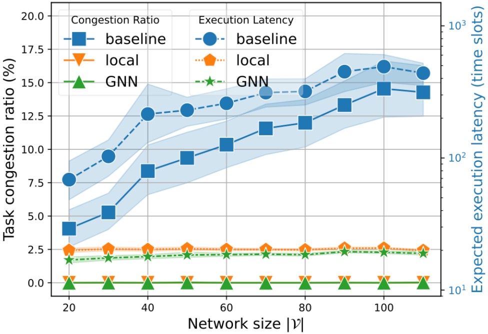


Congestion mitigation in distributed task offloading[15]

Exemplary random test case
on a network of 20 nodes



Barabási–Albert model



If a task is congested, its execution latency > 1000 time slots

Local: all clients can execute their own tasks without congestion
GNN: some tasks offloaded to remote servers without congestion, reducing average execution latency compared to the local policy
Baseline: 4%~15% congestion ratio, and high average execution latency (500)

[15] 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.

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, link scheduling, and routing 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, link scheduling, and routing problems
- 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)

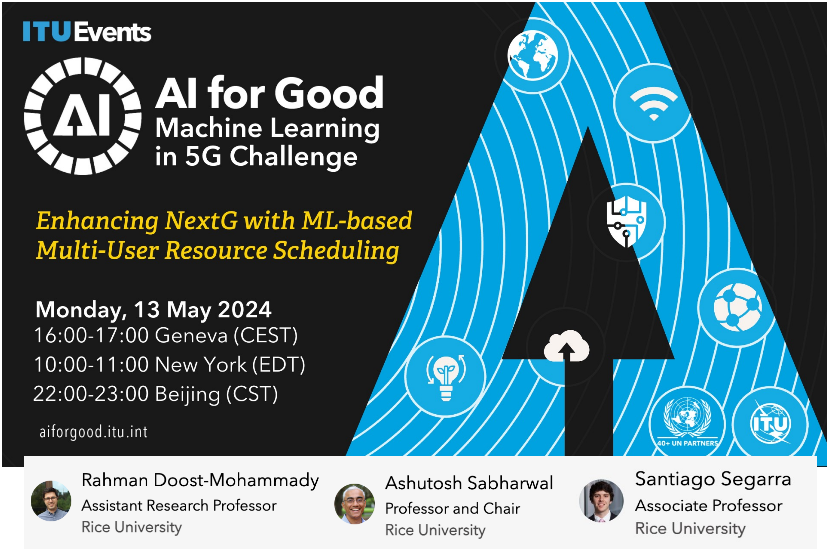
Open and Future Directions

- The 'easy' one $\Rightarrow$ Applications to other problems in wireless (and beyond)
- Implementation in real wireless networks

  $\Rightarrow$ Fading, inexact channel info, packet drops, adversarial/malfunctioning nodes

  $\Rightarrow$ Specific protocols for message passing implementation

Working with Real Data



Open and Future Directions

- The 'easy' one $\Rightarrow$ Applications to other problems in wireless (and beyond)
- Implementation in real wireless networks

  $\Rightarrow$ Fading, inexact channel info, packet drops, adversarial/malfunctioning nodes

  $\Rightarrow$ Specific protocols for message passing implementation

Open and Future Directions

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

Open and Future Directions

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

Open and Future Directions

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

Open and Future Directions

- The 'easy' one $\Rightarrow$ Applications to other problems in wireless (and beyond)
- Implementation in real wireless networks
  - $\Rightarrow$ Fading, inexact channel info, packet drops, adversarial/malfunctioning nodes
  - $\Rightarrow$ Specific protocols for message passing implementation
- Efficient distributed training
  - $\Rightarrow$ New challenges in distributed optimization
  - $\Rightarrow$ Key for implementation in real systems
- Combination with generative AI
  - $\Rightarrow$ 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)