The ~~Un~~reasonable Effectiveness of Graph Neural Networks for Wireless Communications

Jun Zhang

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

In collaboration with

Yifei Shen (MSRA)

Yuanming Shi (ShanghaiTech)

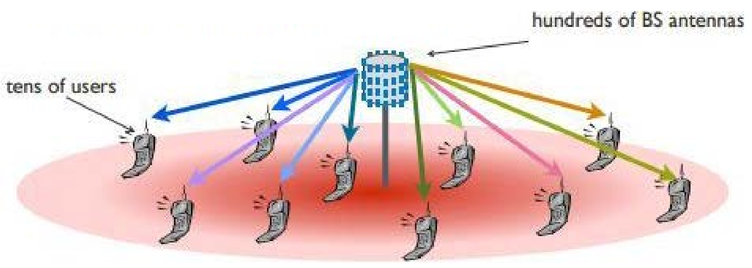
Khaled B. Letaief, S.H. Song (HKUST)

Outline

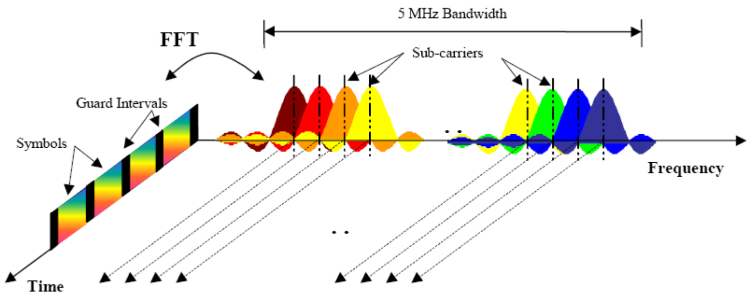
- Computational challenges of wireless networks
- The unreasonable effectiveness of "learn to optimize"
- Graph neural networks (GNNs) for wireless communications
- Theoretical analysis: GNNs vs. MLPs
- Conclusions

Computational challenges of wireless networks

Motivations



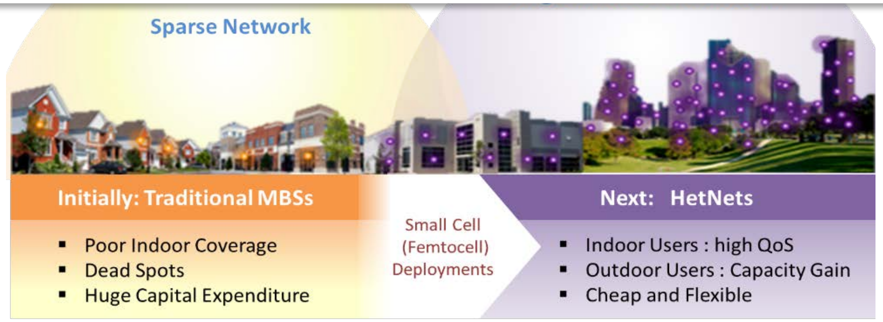
Massive MIMO
(>100 antennas/BS)



OFDM
(>1,000 subcarriers)

Hard large-scale optimization problems are ubiquitous in wireless networks

Massive IoT Access
(millions/km^2)



Ultra-dense Networks
(BS density ~ user density)

Computational challenges in optimizing wireless networks

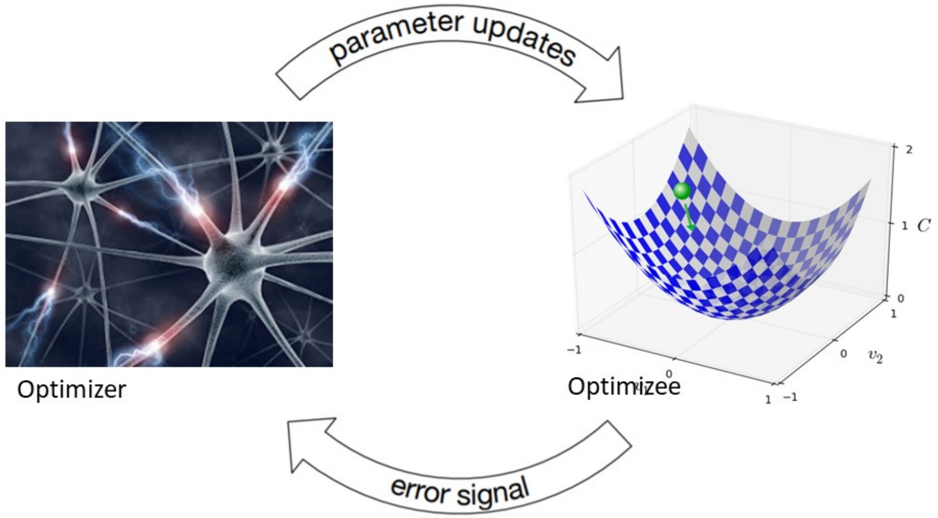
- Enormous problem scales
- Hardness (nonconvexity) of typical problems
- Time-varying channels/topologies (fading/shadowing, mobility)
- Real-time requirement (e.g., AR/VR, UAVs)

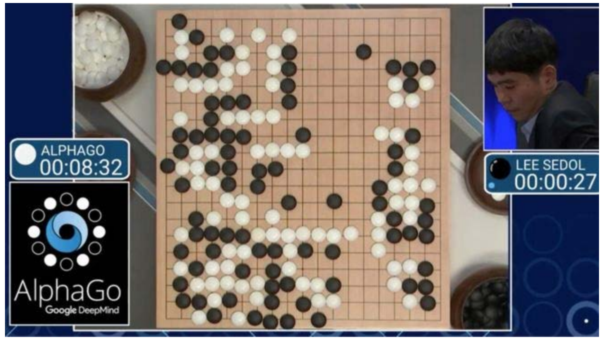
Classic algorithmic approaches

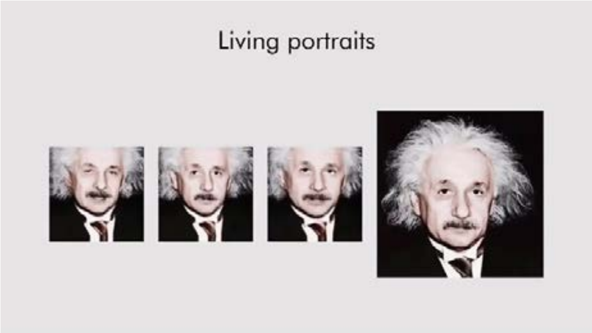
- **Global optimization algorithms**
  - exponential time complexity
  - only work for very small problems

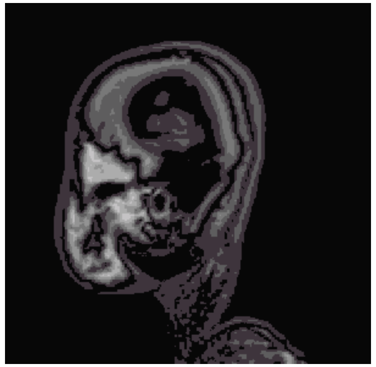
- **Heuristic algorithms**
  - examples: greedy algorithm for user selection
  - hard to design good ones
  - non-negligible gap to the optimal solution
  - difficult to meet real-time requirement

The unreasonable effectiveness of "learn to optimize"





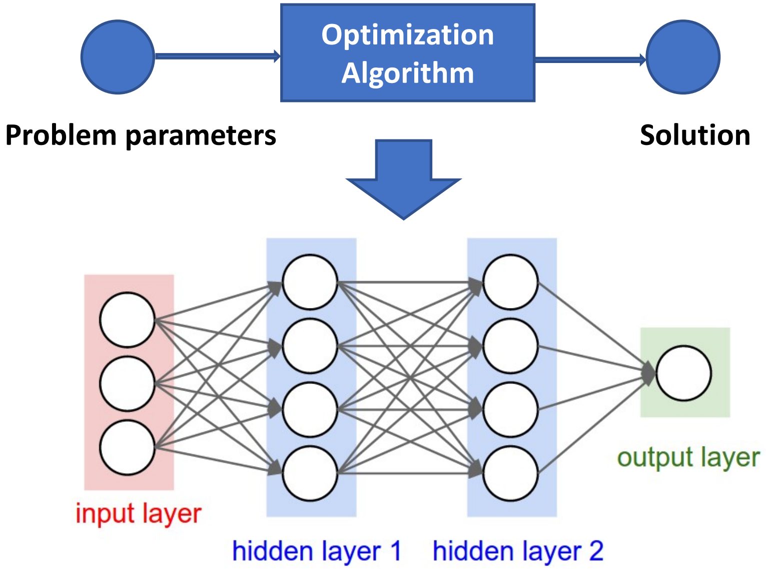




Successes of Deep Learning

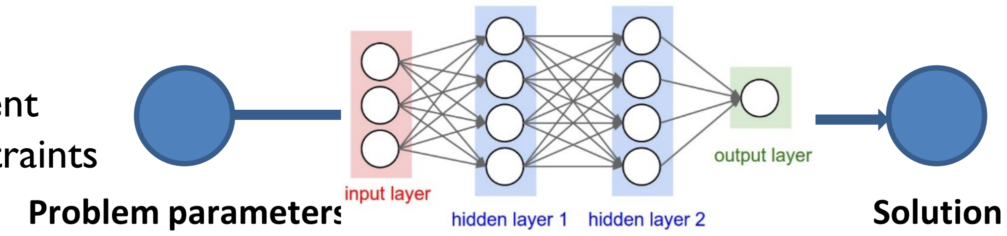
Learning to Optimize

- **Observations**
  - Past data and future problem data have the same distribution
  - Deep learning can learn a good algorithm from data
- **Design Goal**
  - Automatically learn a real-time near-optimal algorithm for difficult optimization problems

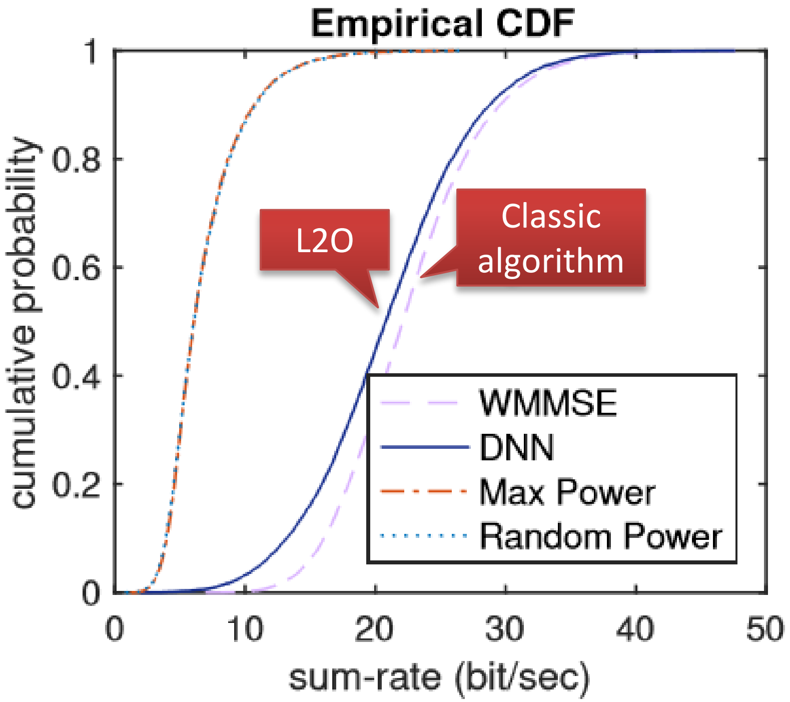


An early attempt

Input
- Channel state
- QoS requirement
- Resource constraints



Output
- Resource allocation
- Detection results



[SCS+18]

- Sum rate maximization of interference channel (NP hard).
- A 3-layer multilayer perceptron (MLP) with [200,80,80] neurons.

[SCS+18] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Trans. Signal Process.*, vol. 66, pp. 5438 – 5453, Oct. 2018.

Limitations of L2O via MLPs

- Poor scalability

# of users (K)	average sum-rate (bit/sec.) DNN	WMMSE	DNN/WMMSE
10	2.770	2.817	98.33%
20	3.363	3.654	92.04%
30	3.498	4.150	84.29%

- Huge amounts of samples
  - Millions of samples;
  - Optimal labels are difficult to generate.
- Weak generalization
  - The output dimension of neural networks must be fixed.

Our targets

High sample efficiency

- To be trained with thousands of unlabelled samples

Scalability

- Be able to work for large scale problems

Good generalization

- Be able to generalize to different problem sizes

Theory-guided design principles

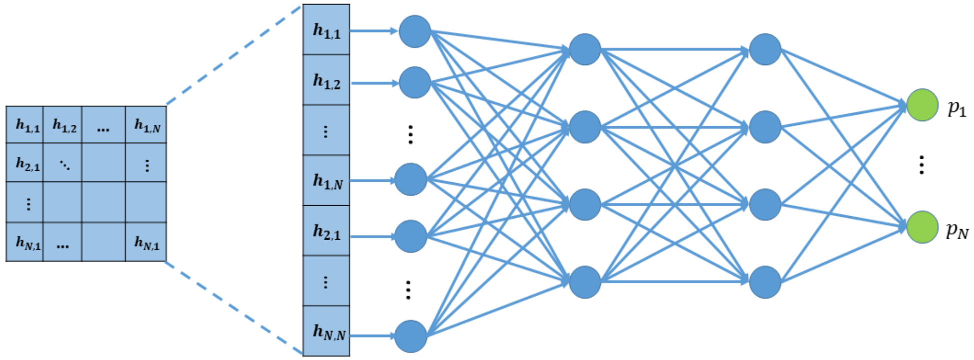
- Effective in designing good neural architecture

Deep Learning: Alchemy or Science?

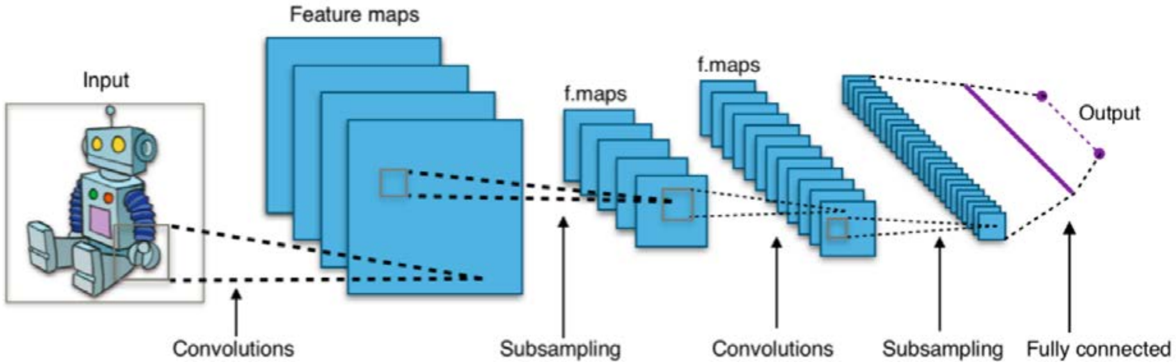
GNNs for wireless communications

Which neural architecture to use?

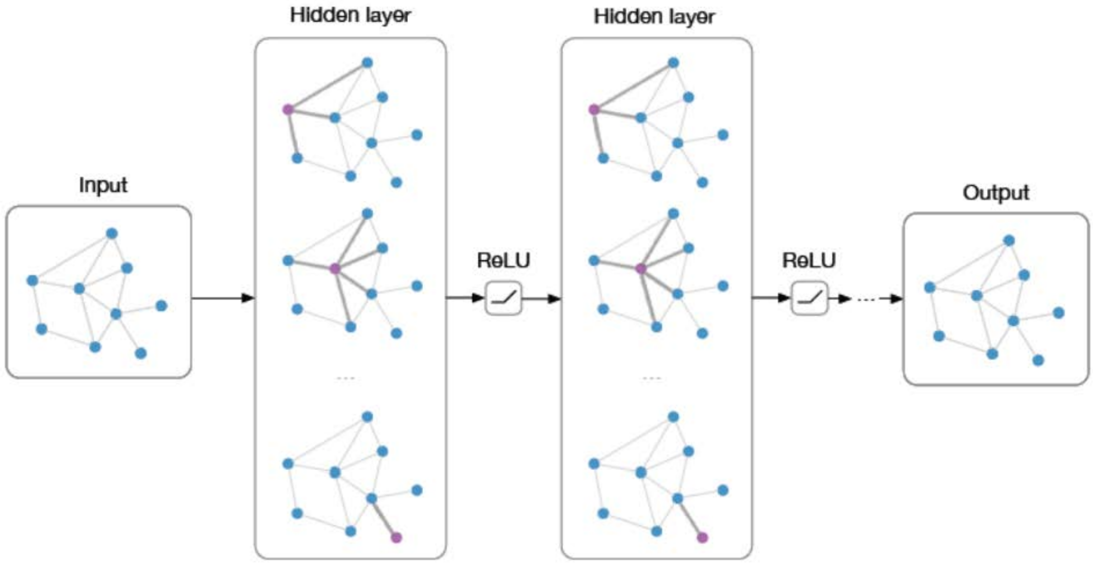
- Why MLP is not effective?
  - It could not exploit structure information in data.



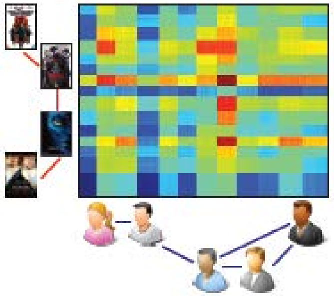
- **A successful story**: Convolutional neural network (CNN) for image processing.
  - It exploits the shift-invariance, local connectivity, compositionality of images.



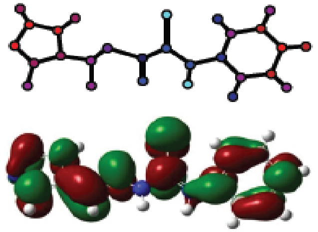
A new architecture: Graph Neural Networks (GNNs)



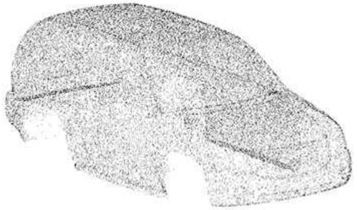
- **GNN Applications**



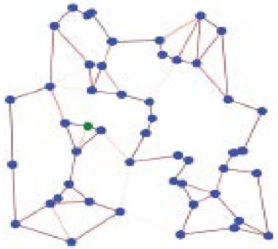
Recommendation



Chemistry



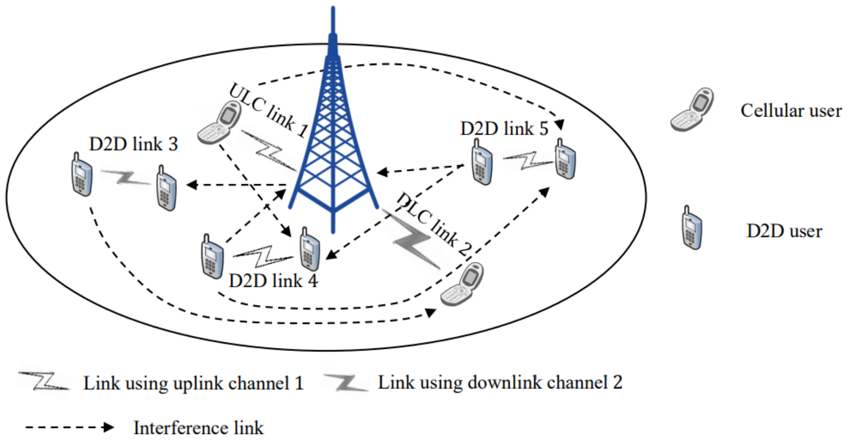
Point clouds

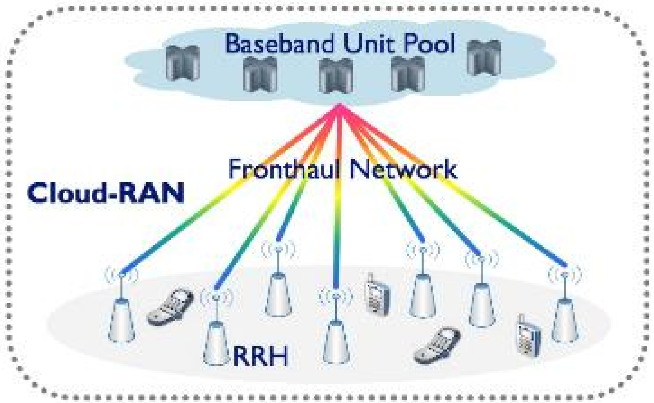


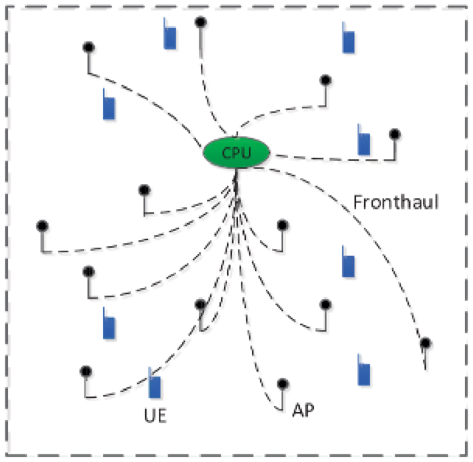
Graph problems

Motivations

- **Observation:** Wireless networks naturally graphs
  - Optimization problems in wireless communication are graph optimization
- **Basic Idea:** Incorporate graph topology into neural networks

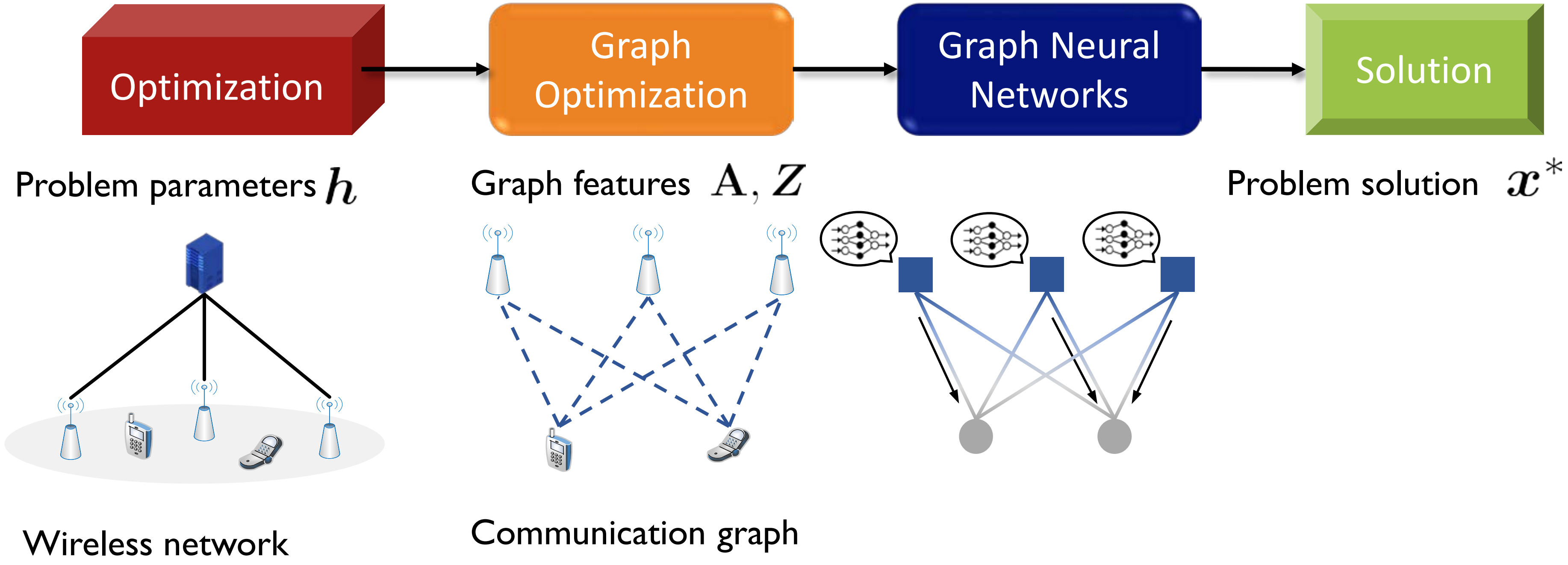






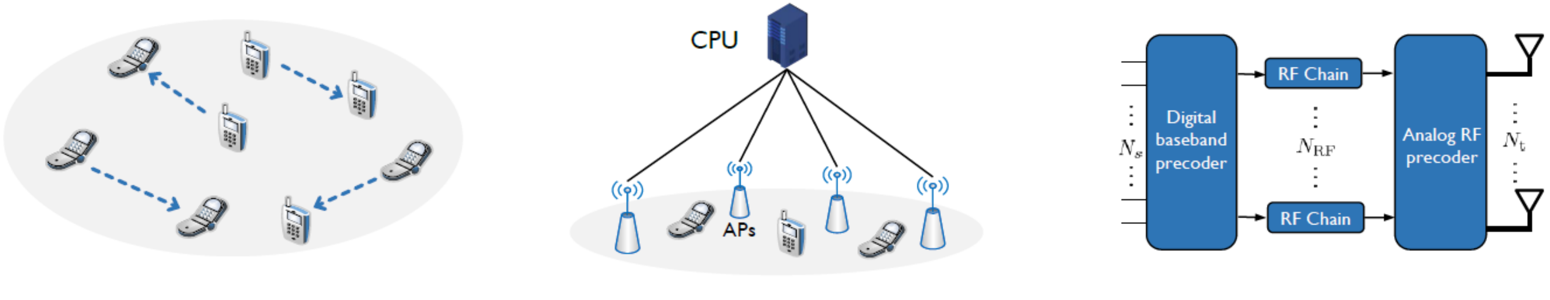
GNN-based framework

- **Proposal:** A two-stage approach for large-scale network optimization

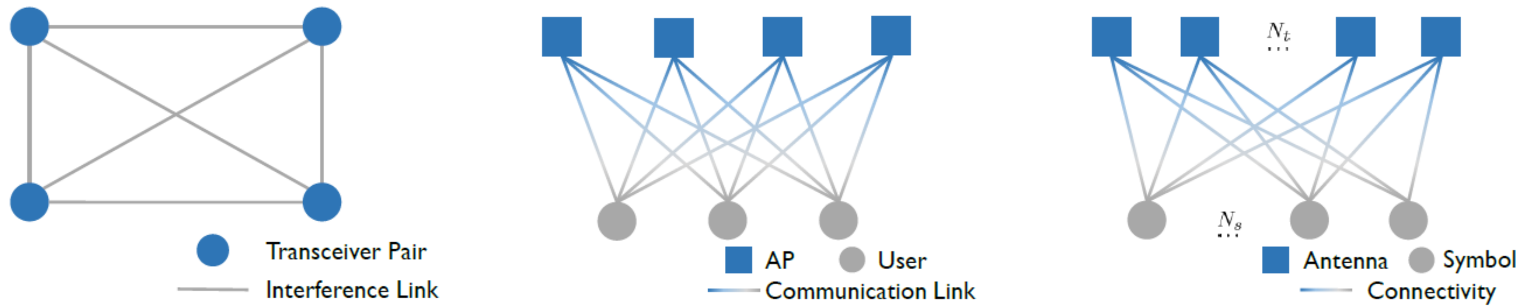


Stage I: Graph modeling

- **Main idea:** network topology as graphs
  - AP/UE as nodes;
  - Communication links as edges
  - Channel information as edge features
- Typical examples: 1) user admission; 1) power control; 2) (hybrid) beamforming

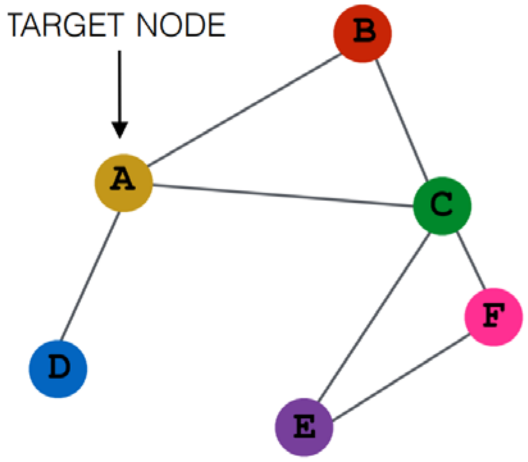


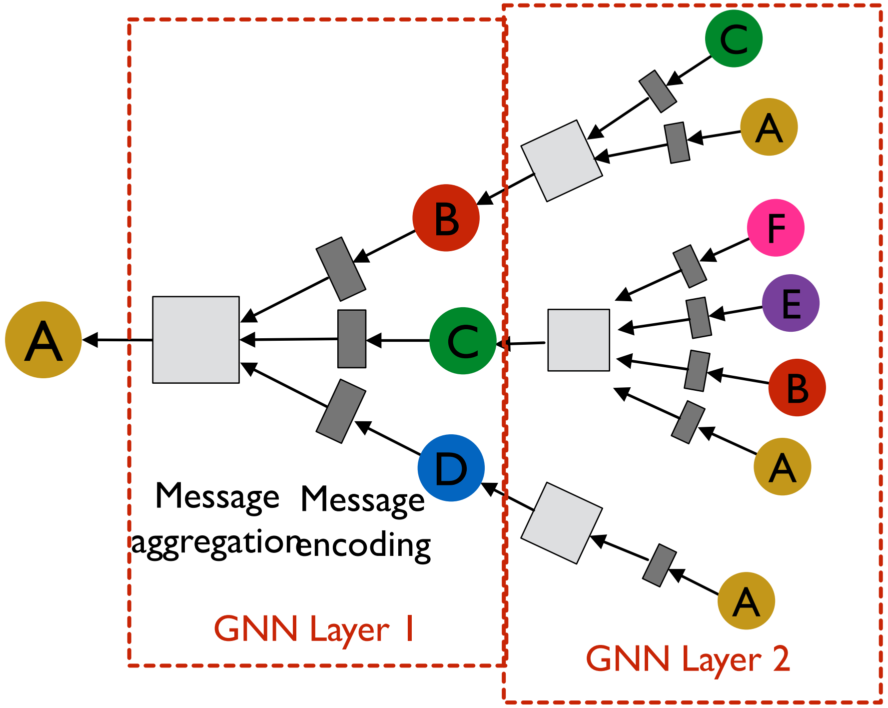
(a) D2D networks. (b) Cell-free networks. (c) Hybrid precoding.



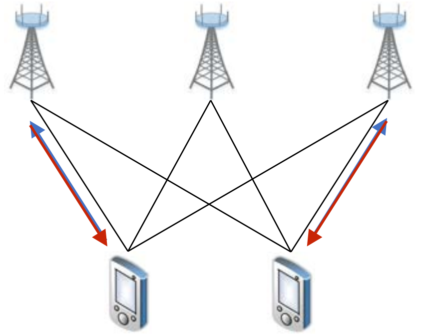
Stage II: Graph neural networks

- **Key idea:** neural message-passing between nodes
- **Two stages:** message encoding & message aggregation





GNNs for wireless networks



($\boldsymbol{V}^{(l)}, \boldsymbol{P}^{(l)}, \boldsymbol{S}^{(l)}, \boldsymbol{U}^{(l)}, \boldsymbol{Q}^{(l)}, \boldsymbol{T}^{(l)}$) are learnable weight matrices

BSs aggregating information from UEs

Learned UE representation

CSI

Message $\mathbf{m}^{(l)}_{\mathrm{UE}\to\mathrm{BS}} = \left(\mathbf{V}^{(l)}\mathbf{u}_j^{(l-1)} + \mathbf{P}^{(l)}\mathbf{h}_{ij}\right)$

Aggregate $\mathbf{b}^{(l)}_{\mathrm{BS}} = \sigma\left(\mathbf{S}^{(l)}\mathbf{b}^{(l-1)}_{\mathrm{BS}} + \sum_{i=1}^{M}\mathbf{m}^{(l)}_{i\to\mathrm{BS}}\right)$

Learned BS representation vector

UEs aggregating information from BSs

Learned BS representation

CSI

Message $\mathbf{m}^{(l)}_{\mathrm{BS}\to\mathrm{UE}} = \left(\mathbf{U}^{(l)}\mathbf{b}_j^{(l-1)} + \mathbf{Q}^{(l)}\mathbf{h}_{ji}\right)$

Aggregate $\mathbf{u}^{(l)}_{\mathrm{UE}} = \sigma\left(\mathbf{T}^{(l)}\mathbf{u}^{(l-1)}_{\mathrm{UE}} + \sum_{i=1}^{K}\mathbf{m}^{(l)}_{i\to\mathrm{UE}}\right)$

Learned UE representation vector

Why GNN? – Theoretical support

- A class of distributed algorithms called *distributed message passing (DMP)*, including many classic algorithms
  - Fractional programming for power control [Shen18TSP]
  - WMMSE for beamforming [Shi11TSP]
  - Riemannian gradient for hybrid precoding [Yu16JSTSP]

- **Theorem:** equivalence between GNNs and DMP algorithms
  1. GNNs are special cases of DMP algorithms
  2. For any DMP algorithm, there exists a GNN that approximates it well

Why GNN? – Theoretical support

- **Theorem:** equivalence between GNNs and DMPs
  1. GNNs are special cases of DMPs
  2. For any DMP algorithm, there exists an GNN that approximate it well

- **Interpretations:**
  1. The distributed property allows GNNs to achieve:
     - **Good generalization**: To generalize to any number of UEs/APs during the test
     - **High computation efficiency**: To have constant running time independent of number of UEs/APs
  2. If a GNN is trained well, its performance is at least as good as DMPs

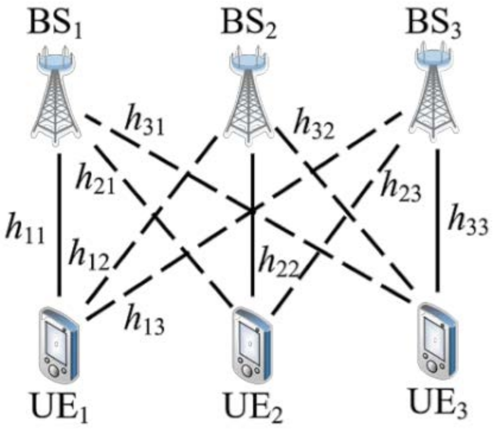
Case study: Power control

- **Sum rate maximization in K-user interference channel**
  - h_{kk} is the direct channel of k-th user
  - h_{kj} is the interference channel between j-th and k-th user
  - p_k is the power for k-th transmitter

- **NP-hard**

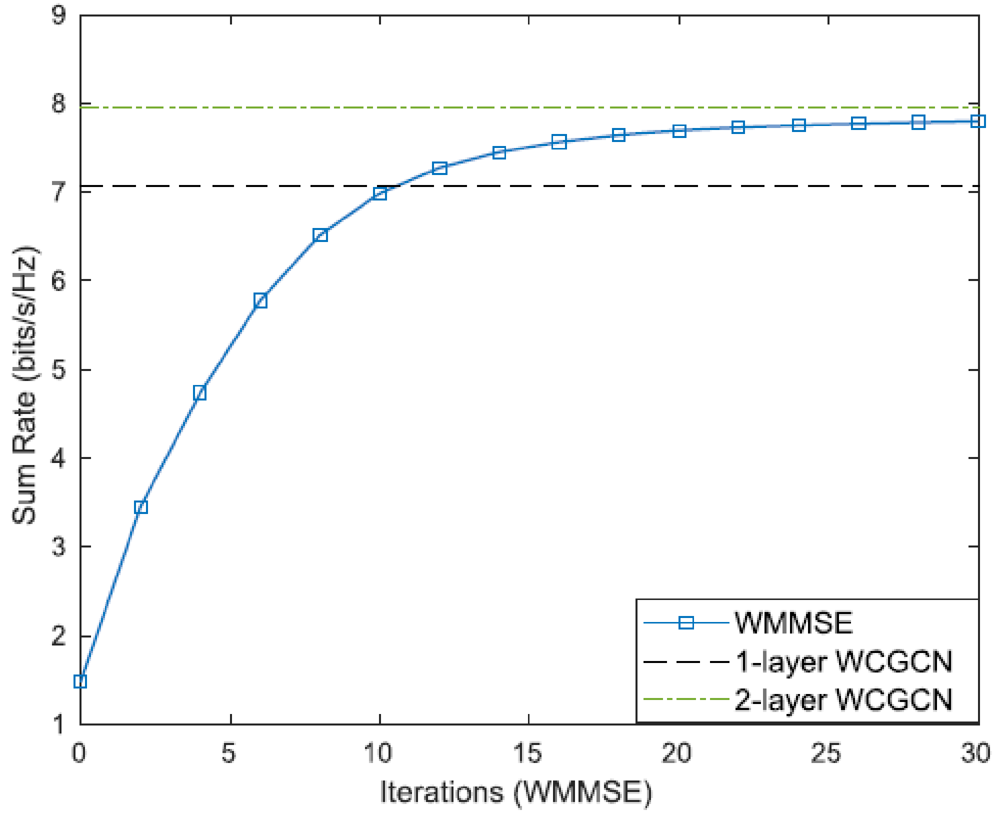
$$\begin{aligned} \underset{p_1,\cdots,p_K}{\text{maximize}} \quad & \sum_{k=1}^{K} \log_2\left(1+\frac{|h_{kk}|^2 p_k}{\sum_{j\neq k}|h_{kj}|^2 p_j+\sigma_k^2}\right) \\ \text{subject to} \quad & 0 \le p_k \le 1. \end{aligned}$$

We adopt unsupervised training, i.e., no label is needed!



Simulations: GNN vs. Distributed Algorithm

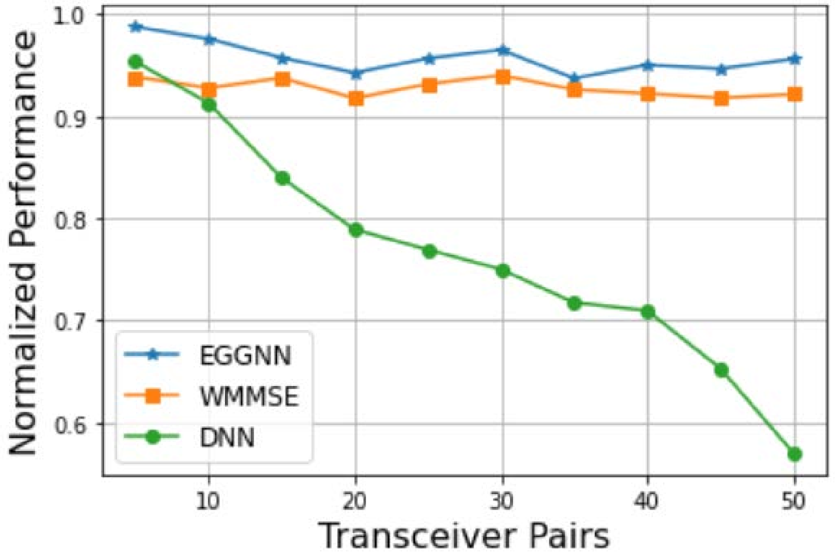
- A 1-layer GNN outperforms WMMSE (it belongs to DMP) with 10 iterations.
- A 2-layer GNN outperforms WMMSE with 30 iterations.



Simulations: Improved performance

- **Legends**
  - EGGNN: Proposed GNN method
  - WMMSE: a widely adopted optimization-based method [Shi11TSP]
  - DNN: DNN-based method [Sun18TSP]
- **Advantages**
  - Near-optimal performance, better than WMMSE
  - Much better scalability than DNN

$$\begin{aligned} \underset{\mathbf{p}}{\text{maximize}} \quad & \sum_{k=1}^{K} \log_2 \left(1 + \frac{|h_{k,k}|^2 p_k}{\sum_{j \neq k}^{K} |h_{j,k}|^2 p_j + \sigma_k^2} \right), \\ \text{subject to} \quad & 0 \leq p_k \leq P \end{aligned}$$

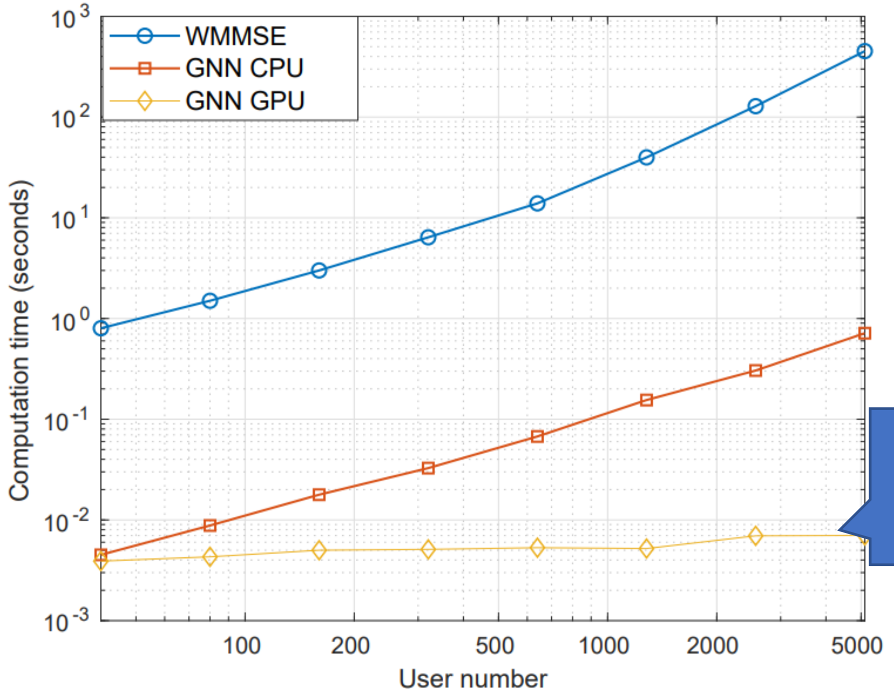


Performance is normalized by the best of FPlinQ with 100 different initialization points

Simulations: Scalability

- **Setting:** GNN trained on 50 users and test on different numbers of users

- **Advantages**
  - Orders of magnitude of speedups
  - On CPU, computation time is reduced by 100~1000 times compared with WMMSE
  - Nearly constant time scale up on GPU



6 ms for thousands of users;
100,000x reduction!

Simulations: Generalization

- **Generalization to larger scales**
  - Trained with 50 pairs in a 1000m×1000m region.
  - Test with different numbers of pairs while the density of users is fixed.

GENERALIZATION TO LARGER PROBLEM SCALES BUT SAME DENSITY

Links	Size (m^2)	($d_{\min}, d_{\max}$)	
		(10m,50m)	(30m,30m)
200	2000×2000	98.3%	98.1%
400	2828×2828	98.9%	98.2%
600	3464×3464	98.8%	98.7%
800	4000×4000	98.9%	98.6%
1000	4772×4772	98.9%	98.7%

- **Generalization to higher densities**
  - Trained with 50 pairs in a 1000m×1000m region.
  - Change the number of pairs in the test set while fixing the area size.

GENERALIZATION OVER DIFFERENT LINK DENSITIES. THE PERFORMANCE LOSS COMPARED TO $K = 50$ IS SHOWN IN THE BRACKET

Links	Size (m^2)	($d_{\min}, d_{\max}$)	
		(10m,50m)	(30m,30m)
100	1000×1000	97.6% (+0.1%)	96.4% (−0.1%)
200		97.0% (−0.5%)	96.0% (−0.5%)
300		95.9% (−1.6%)	94.9% (−1.6%)
400		95.6% (−1.9%)	94.5% (−2.0%)
500		95.3% (−2.2%)	94.5% (−2.0%)

Theoretical analysis: GNN vs. MLP

The reasonable effectiveness of GNNs

Theoretical analysis

- So far, we have shown
  - Incorporating prior knowledge of wireless communications into neural architectures improves performance

- **Goal of this part:** to theoretically characterize
  - *How many training samples* are needed to train the neural network well?
  - *How much performance gains* can "structure" bring?

Generalization analysis of deep learning

- **Generalization Analysis via PAC-learning**
  - Test data are drawn i.i.d. from unknown distribution *D;*
  - $f^*(\cdot)$ is an oracle algorithm generating an optimal solution;
  - $f(x_i, W^*)$ is neural network where *W* is weight obtained by training;
  - Given *N* training samples, if with probability at least δ, we have

$$\mathbb{E}_{x\sim\mathcal{D}}\|f(x, W^*) - f^*(x)\| \leq \epsilon$$

  - then we call $f^*(\cdot)$ is (ε, δ, N) learnable by this neural network.
  - The minimum number of training samples *N* is called the sample complexity

- **Key idea (algorithmic alignment)**
  - If the neural network shares a common structure with the oracle function $f^*(\cdot)$, then the sample complexity is low.

Keyulu Xu, Jingling Li, et. al, "What can neural networks reason about?" ICLR 2020.

Main result

- Target Example: GNNs versus unstructured deep neural networks (DNNs), i.e., MLP, for "learning to optimize" to solve large-scale graph optimization problems.

- **Theorem:** Consider an optimization problem on a |V|-node graph.
  - For GNNs, it is $(\epsilon, \delta, \mathcal{O}(C_{\mathcal{A}}(f_0, \cdots, f_m)/|V|))$-learnable.
  - For MLPs, it is $(\epsilon, \delta, \mathcal{O}(C_{\mathcal{A}}(f_0, \cdots, f_m)|V|))$-learnable.
    - $C_{\mathcal{A}}(f_0, \cdots, f_m)$ is a constant related to
      1. maximum degree of the graph;
      2. optimization objectives;
      3. NN substructure, e.g., activation.

Proof sketch

- **Lemma 1:** For any graph optimization problem, there exists a distributed message passing algorithm that can solve it.
- **Lemma 2:** If the neural network *A* can simulate the task algorithm with *n* modules, and each module is $(\varepsilon, \delta, M/n)$ learnable, then the task is (ε, δ, M) learnable by *A*.
- **Lemma 3:** To simulate a DMP algorithm,
  - GNNs require $\mathcal{O}(1)$ modules that are $(\epsilon, \delta, O(C/|V|))$ learnable;
  - MLPs require $\mathcal{O}(|V|)$ modules that are $(\epsilon, \delta, O(C))$ learnable.

GNNs aligns with DMP algorithms better than MLPs

Corollary: GNNs vs. MLPs

- To learn DMP algorithms to solve a graph optimization problem.
  - For GNNs, it is $(\epsilon, \delta, \mathcal{O}(C_{\mathcal{A}}(\psi, f_0, \cdots, f_m)/|V|))$-learnable
  - For MLPs, it is $(\epsilon, \delta, \mathcal{O}(C_{\mathcal{A}}(\psi, f_0, \cdots, f_m)|V|))$-learnable

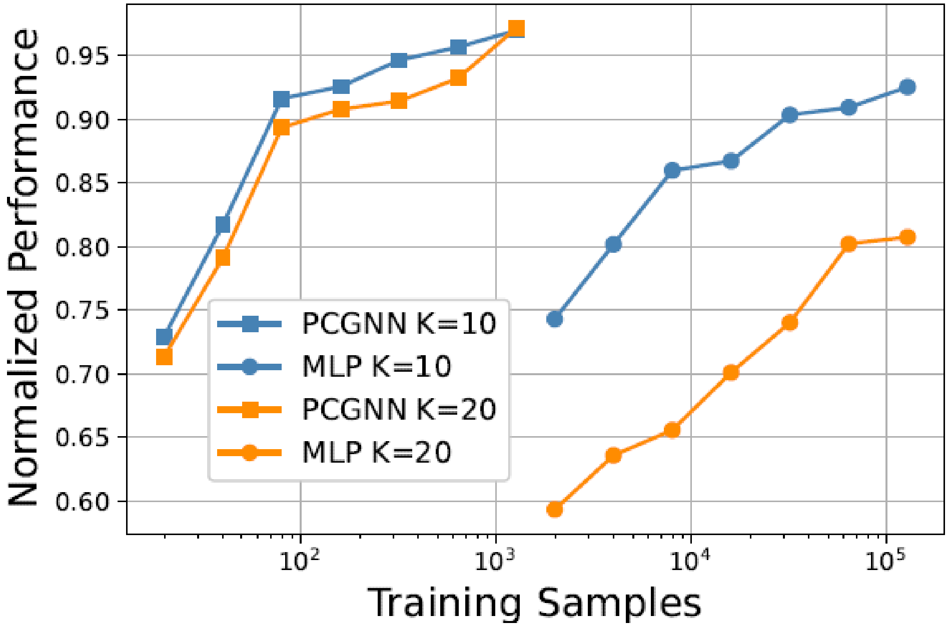
→ GNNs vs. MLPs (sample complexity)

- GNNs require $\mathcal{O}(|V|^2)$ times fewer training samples than MLPs.
- GNNs' performance gap to the optimal solution is $\mathcal{O}(|V|)$ times lower than MLPs (see our paper for more details).

Simulations: Sample complexity

- We consider K-user interference channel power control.
- **Theory:** MLPs require $\mathcal{O}(|V|^2)$ times more training samples than GNNs.
- **Simulations**
  - For 10 users, MLPs require 100 times more training samples;
  - For 20 users, MLPs require 400~600 times more training samples.

$$\begin{aligned} \underset{\boldsymbol{p}}{\text{maximize}} \quad & \sum_{k=1}^{K} \log_2 \left(1 + \frac{|h_{k,k}|^2 p_k}{\sum_{j \neq k}^{K} |h_{j,k}|^2 p_j + \sigma_k^2} \right), \\ \text{subject to} \quad & 0 \leq p_k \leq P \end{aligned}$$



Performance is normalized by the best of FPlinQ with 100 different initialization points

Corollary: Stable performance of GNNs

- Consider a graph optimization problem on graph $G = (V, E)$
  - For GNNs, it is $(\epsilon, \delta, \mathcal{O}(C_{\mathcal{A}}(\psi, f_0, \cdots, f_m)/|V|))$-learnable
    - $C_{\mathcal{A}}(\psi, f_0, \cdots, f_m)$ is a constant related to 1) maximum degree of the graph; 2) optimization objectives; 3) NN substructure, e.g., activation.

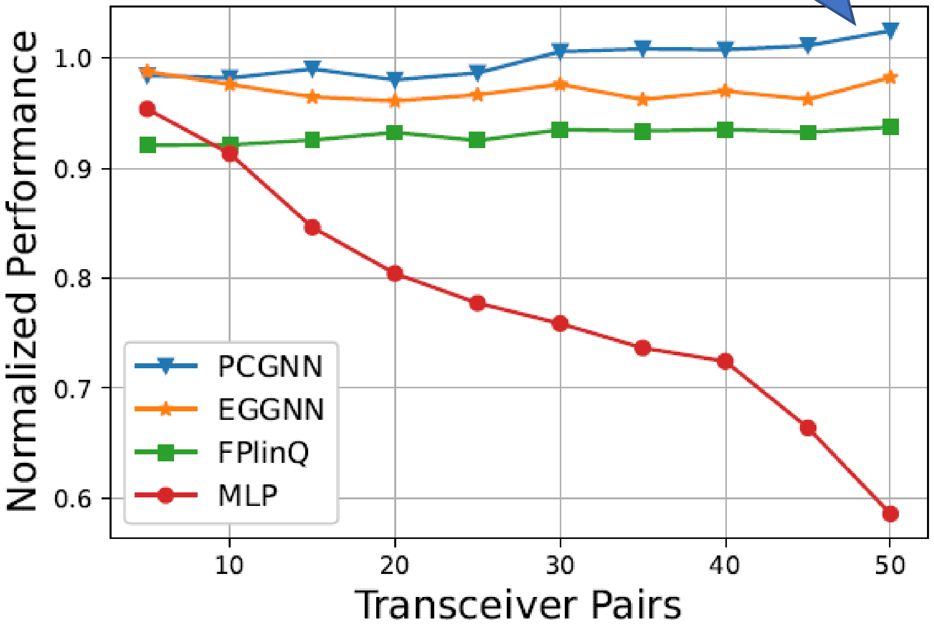
Stable performance of GNNs

- If the maximum degree does not change, the performance of GNNs is stable and independent of the node (user) number;
- Neural architectures can be specially designed to improve the bound.

Simulations: Neural architecture design

- **Legends**
  - **PCGNN**: GNN designed to optimize the bound
  - EGGNN: [Shen20JSAC]
  - FPlinQ: optimization-based method (fractional programming)
  - MLP: MLP-based method [Sun18TSP]
- **Simulations**
  - PCGNN shows consistently better performance than ECGNN.

PCGNN outperforms Best FPlinQ (among 100 trials)



Performance is normalized by the best of FPlinQ with 100 different initialization points

Conclusions

Conclusions

- GNNs for wireless communications
  - Good performance (beat SOTA algorithms)
  - Good generalization (to different problem sizes)
  - High computational efficiency (orders of magnitude speedup)
  - Wide applications
- The **reasonable** effectiveness of GNNs
  - The common structure in the target task and neural network improves sample efficiency and performance
  - Supported quantitively via PAC-learning theory + algorithmics alignment
- For reproducibility
  - https://github.com/yshenaw/GNN4Com

Future directions

- **The picture is far from complete!**
- Extend to other applications
  - GNNs for channel estimation
  - GNNs for MIMO detection
  - …
- Extend to other learning approaches
  - GNNs with model-driven deep learning
- Further improve robustness to distribution shift
  - To improve out of distribution (OOD) generalization

ROBUSTNESS TO THE CHANGE OF CHANNELS. $K = 10$.

Setting	User distribution shift	Antenna height distribution shift	LoS $\rightarrow$ NLoS	ITU $\rightarrow$ LTE
PCGNN	97.62%	96.90%	94.43%	88.71%
PCGNN (Full Training)	97.78%	97.63%	97.53%	96.57%
FPlinQ	93.51%	93.66%	93.51%	94.32%

References

- H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Trans. Signal Process.*, vol. 66, pp. 5438 – 5453, Oct. 2018.
- Keyulu Xu, Jingling Li, et. al, "What can neural networks reason about?" ICLR 2020.
- Y. Shen, **J. Zhang**, S.H. Song, and K. B. Letaief, "Graph neural networks for wireless communications: From theory to practice," submitted to *IEEE Trans. Wireless Communications*.
- Y. Shen, Y. Shi, **J. Zhang**, and K. B. Letaief, "Graph neural networks for scalable radio resource management: architecture design and theoretical analysis," *IEEE J. Select. Areas Commun.*, vol. 39, no. 1, pp. 101–115, Jan. 2021.
- Y. Shen, **J. Zhang**, and K. B. Letaief, "How neural architectures affect deep learning for communication networks?" *IEEE Int. Conf. Commun. (ICC)*, Seoul, South Korea, May 2022.

Thank you!

- For more details

  https://eejzhang.people.ust.hk/