

# Graph Neural Networks for Scalable Radio Resource Management: Architecture Design and Theoretical Analysis

Yifei Shen, *Student Member, IEEE*, Yuanming Shi, *Member, IEEE*, Jun Zhang, *Senior Member, IEEE*, and Khaled B. Letaief, *Fellow, IEEE*

## Abstract

Deep learning has recently emerged as a disruptive technology to solve challenging radio resource management problems in wireless networks. However, the neural network architectures adopted by existing works suffer from poor scalability, generalization, and lack of interpretability. A long-standing approach to improve scalability and generalization is to incorporate the structures of the target task into the neural network architecture. In this paper, we propose to apply graph neural networks (GNNs) to solve large-scale radio resource management problems, supported by effective neural network architecture design and theoretical analysis. Specifically, we first demonstrate that radio resource management problems can be formulated as graph optimization problems that enjoy a universal permutation equivariance property. We then identify a class of neural networks, named *message passing graph neural networks* (MPGNNs). It is demonstrated that they not only satisfy the permutation equivariance property, but also can generalize to large-scale problems while enjoying a high computational efficiency. For interpretability and theoretical guarantees, we prove the equivalence between MPGNNs and a class of distributed optimization algorithms, which is then used to analyze the performance and generalization of MPGNN-based methods. Extensive simulations, with power

The materials in this paper were presented in part at the IEEE Global Communications Conference (Globecom) Workshops, 2019 [1]. This work was supported by the General Research Fund (Project No. 16210719) from the Research Grants Council of Hong Kong.

Y. Shen and K. B. Letaief are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong (E-mail: {yshenaw, eekhaled}@ust.hk). Y. Shi is with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (E-mail: shiym@shanghaitech.edu.cn). J. Zhang is with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong (E-mail: jun-eie.zhang@polyu.edu.hk). (The corresponding author is J. Zhang.)

control and beamforming as two examples, will demonstrate that the proposed method, trained in an unsupervised manner with unlabeled samples, matches or even outperforms classic optimization-based algorithms without domain-specific knowledge. Remarkably, the proposed method is highly scalable and can solve the beamforming problem in an interference channel with 1000 transceiver pairs within 6 milliseconds on a single GPU.

### **Index Terms**

Radio resource management, wireless networks, graph neural networks, distributed algorithms, permutation equivariance.

## **I. INTRODUCTION**

Radio resource management, e.g., power control [2] and beamforming [3], plays a crucial role in wireless networks. Unfortunately, many of these problems are non-convex and computationally challenging. Moreover, they need to be solved in a real-time manner given the time-varying wireless channels and the latency requirement of many mobile applications. Great efforts have been put forward to develop effective algorithms for these challenging problems. Existing algorithms are mainly based on convex optimization approaches [4], [5], which have a limited capability in dealing with non-convex problems and scale poorly with the problem size. Problem specific algorithms can be developed, which, however, is a laborious process and requires much problem specific knowledge.

Inspired by the recent successes of deep learning in many application domains, e.g., computer vision and natural language processing [6], researchers have attempted to apply deep learning based methods, particularly, “learning to optimize” approaches, to solve difficult optimization problems in wireless networks [7]–[15]. The goal of such methods is to achieve near-optimal performance in a real-time manner without domain knowledge, i.e., to automate the algorithm design process. There are two common paradigms on this topic [16], [17]. The first one is “end-to-end learning”, which directly employs a neural network to approximate the optimal solution of an optimization problem. For example, in [7], to solve the power control problem, a multi-layer perceptron (MLP) was used to approximate the input-output mapping of the classic weighted minimum mean square error (WMMSE) algorithm [18] to speed up the computation. The second paradigm is “learning alongside optimization”, which replaces some ineffective policy in a traditional algorithm with a neural network. For example, an MLP was utilized in [11] to

replace the pruning policy in the branch-and-bound algorithm. Accordingly, significant speedup and performance gain in the access point selection problem was achieved compared with the optimization-based methods in [19], [20].

A key design ingredient underlying both paradigms of “learning to optimize” is the neural network architecture. Most of the existing works adopt MLPs [7], [9], [11], [21] or convolutional neural networks (CNNs) [8], [12]. These architectures are inherited from the ones developed for image processing tasks and thus are not tailored to problems in wireless networks. Although near-optimal performance is achieved for small-scale wireless networks, they fail to exploit the wireless network structure and thus suffer from poor scalability and generalization in large-scale radio resource management problems. Specifically, the performance of these methods degrades dramatically when the wireless network size becomes large. For example, it was shown in [7] that the performance gap to the WMMSE algorithm is 2% when  $K = 10$  and becomes 12% when  $K = 30$ . Moreover, these methods generalize poorly when the number of agents in the test dataset is larger than that in the training dataset. In dense wireless networks, resource management may involve thousands of users simultaneously and the number of users changes dynamically, thus, making the wide application of these learning-based methods very difficult.

A long-standing idea to improve scalability and generalization is to incorporate the structures of the target task into the neural network architecture [16], [21], [22]. A prominent example is the development of CNNs for computer vision, which is inspired by the fact that the neighbor pixels of an image are useful when they are considered together [23]. To achieve better scalability, structures in a single-antenna system with homogeneous agents have recently been exploited for effective neural network architecture design [10], [14]. In static channels, observing that channel states are deterministic functions of users’ geo-locations in a 2D Euclidean space, spatial convolution was developed in [10], which is applicable in wireless networks with thousands of users but cannot handle fading channels. With fading channels, it was observed that the channel matrix can be viewed as the adjacency matrix of a graph [14]. From this perspective, a random edge graph neural network (REGNN) operating on such a graph was developed, and it was demonstrated that it inhibits a good generalization property when the number of users in the wireless networks changes. However, in a multi-antenna system or a single-antenna system with heterogeneous agents, the channel matrix no longer fits the form of an adjacency matrix and the REGNN cannot be applied.

In this paper, we address the limitations of existing works by modeling wireless networks as

*wireless channel graphs* and develop neural networks to exploit the graph topology. Specifically, we treat the agents as nodes in a graph, communication channels as directed edges, agent specific parameters as node features, and channel related parameters as edge features. Subsequently, low-complexity neural network architectures operating on wireless channel graphs will be proposed.

Existing works (e.g., [7], [11], [13]) also have another major limitation, namely, they treat the adopted neural network as a black box. Despite the superior performance in specific applications, it is hard to interpret what is learned by the neural networks. To ensure reliability, it is crucial to understand when the algorithm works and when it fails. Thus, a good theoretical understanding is demanded for the learning-based radio resource management methods. Compared with learning-based methods, conventional optimization-based methods are well-studied. This inspires us to build a relationship between these two types of methods. In particular, we shall prove the *equivalence* between the proposed neural networks and a favorable class of optimization-based methods. This equivalence will allow the development of tractable analysis for the performance and generalization of the learning-based methods through the study of their equivalent optimization-based methods.

#### A. Contributions

In this paper, we develop scalable learning-based methods to solve radio resource management problems in dense wireless networks. The major contributions are summarized as follows:

- 1) We model wireless networks as wireless channel graphs and formulate radio resource management problems as graph optimization problems. We then show that a permutation equivariance property holds in general radio resource management problems, which can be exploited for effective neural network architecture design.
- 2) We identify a favorable class of neural networks operating on wireless channel graphs, namely MPGNNs. In such neural networks, the feature of each node is updated by aggregating information from local nodes and edges with a low-complexity permutation invariant function. Thus, MPGNNs satisfy the permutation equivariance property, and have the ability to generalize to large-scale problems while enjoying a high computational efficiency.
- 3) For an effective implementation, we propose a *wireless channel graph convolution network* (WGCN) within the MPGNN class. Besides inheriting the advantages of MPGNNs, the WGCN enjoys several unique advantages for solving radio resource management problems. First, it can effectively exploit both agent-related features and channel-related

features effectively. Second, it is insensitive to the corruptions of features, e.g., channel state information (CSI), implying that they can be applied with partial and imperfect CSI.

- 4) To provide interpretability and theoretical guarantees, we prove the equivalence between MPGNNs and a class of distributed optimization algorithms, which include many classic algorithms for radio resource management, e.g., WMMSE [18]. Based on this equivalence, we analyze the performance and generalization of MPGNN-based methods in the weighted sum rate maximization problem.
- 5) We test the effectiveness of WCGCN for power control and beamforming problems, training with unlabeled data. Extensive simulations will demonstrate that the proposed WCGCN matches or outperforms classic optimization-based algorithms without domain knowledge, and with significant speedups. Remarkably, WCGCN can solve the beamforming problem with 1000 users within 6 milliseconds on a single GPU.<sup>1</sup>

## B. Notations

Throughout this paper, superscripts  $(\cdot)^H$ ,  $(\cdot)^T$ ,  $(\cdot)^{-1}$  denote conjugate transpose, transpose, inverse, respectively. The set symbol  $\{\}$  in this paper denotes a multiset. A multiset is a 2-tuple  $X = (S, m)$  where  $S$  is the underlying set of  $X$  that is formed from its distinct elements, and  $m : S \rightarrow \mathbb{N}_{\geq 1}$  gives the multiplicity of elements. For example,  $\{a, a, b\}$  is a multiset where element  $a$  has multiplicity 2 and element  $b$  has multiplicity 1.

## II. GRAPH MODELING OF WIRELESS NETWORKS

In this section, we model wireless networks as graphs, and formulate radio resource management problems as graph optimization problems. Key properties of radio resource management problems will be identified, which will then be exploited to design effective neural network architectures.

### A. Directed Graphs and Permutation Equivariance Property

A directed graph can be represented as an ordered pair  $G = (V, E)$ , where  $V$  is the set of nodes and  $E$  is the set of edges. Node  $x$  is *adjacent* to node  $y$  if  $(x, y) \in E$ , denoted as  $x \sim y$ . Two graphs  $G = (V, E)$  and  $G' = (V', E')$  are *isomorphic* if there is a bijection  $f : V \rightarrow V'$

<sup>1</sup>The codes to reproduce the simulation results will be made available soon.

such that  $u \sim v \iff f(u) \sim f(v)$ , denoted by  $G \cong G'$ . The *adjacency matrix* of  $G$  is an  $n \times n$  matrix  $\mathbf{A} \in \{0, 1\}^{n \times n}$ , where  $\mathbf{A}_{i,j} = 1$  if and only if  $(i, j) \in E$  for all  $i, j \in V$ . A directed graph  $G$  can be represented as an adjacency matrix. The permutation  $\pi : \{0, 1, \dots, n\} \rightarrow \{0, 1, \dots, n\}$  corresponds to a permutation matrix  $\mathbf{P}$ . The rows (or columns) of  $\mathbf{A}$  are rearranged if  $\mathbf{P}$  is left (or right) multiplied to  $\mathbf{A}$ . The matrix  $\mathbf{PAP}^T$  is also an adjacency matrix. Graphs corresponding to adjacency matrices  $\mathbf{A}$  and  $\mathbf{PAP}^T, \forall \mathbf{P}$  are isomorphic since applying the permutation is a re-ordering of nodes, denoted by  $\mathbf{A} \cong \mathbf{PAP}^T$ .

We now introduce optimization problems defined on directed graphs, and identify their permutation invariance and equivariance properties. We assign each node  $v_i \in V$  an optimization variable  $\gamma_i \in \mathbb{R}^n$ . Denote  $\mathbf{\Gamma} = [\gamma_1, \dots, \gamma_{|V|}]$ , then an optimization problem defined on graph  $G$  can be written as

$$\mathcal{Q} : \underset{\mathbf{\Gamma}}{\text{minimize}} \quad g(\mathbf{\Gamma}, \mathbf{A}) \quad \text{subject to} \quad Q(\mathbf{\Gamma}, \mathbf{A}) \leq 0, \quad (1)$$

where  $g(\cdot, \cdot)$  represents the objective function and  $Q(\cdot, \cdot)$  represents the constraint.

As  $\mathbf{A} \cong \mathbf{PAP}^T$ , optimization problems defined on graphs have the permutation invariance property as stated below.

**Proposition II.1.** (*Permutation invariance*) *For any permutation matrix  $\mathbf{P}$ , the optimization problem defined in (1) has the following property*

$$g(\mathbf{\Gamma}, \mathbf{A}) = g(\mathbf{\Gamma P}^T, \mathbf{PAP}^T), \quad Q(\mathbf{\Gamma}, \mathbf{A}) = Q(\mathbf{\Gamma P}^T, \mathbf{PAP}^T).$$

*Proof.* Since adjacency matrices  $\mathbf{A}$  and  $\mathbf{PAP}^T$  represent the same graph, permuting  $\mathbf{\Gamma}$  and  $\mathbf{A}$  simultaneously is simply a reordering of the variables. As a result, we have  $g(\mathbf{\Gamma}, \mathbf{A}) = g(\mathbf{\Gamma P}^T, \mathbf{PAP}^T)$  and  $Q(\mathbf{\Gamma}, \mathbf{A}) = Q(\mathbf{\Gamma P}^T, \mathbf{PAP}^T)$ .  $\square$

The permutation invariance property of the objective value and constraint leads to the corresponding property of sublevel sets. We first define the sublevel sets.

**Definition II.1.** (Sublevel sets) The  $\alpha$  sublevel set of a function  $f : \mathbb{C}^n \rightarrow \mathbb{R}$  is defined as

$$R^\alpha = \{x \in \text{dom} f \mid f(x) \leq \alpha\},$$

where  $\text{dom} f$  is the feasible domain.

Denote the optimal objective value of (1) as  $z^*$ , and the set of  $\epsilon$ -accurate solutions as  $\mathcal{R}^{z^*+\epsilon}$ . Thus, the properties of sublevel sets imply the properties of near-optimal solutions. Specifically, the permutation invariance property of the objective function implies the permutation equivariance property of the sub-level sets, which is stated in the next proposition.

**Proposition II.2.** (*Permutation equivariance*) Denote  $\mathcal{R}^\alpha$  as the sublevel set of  $g(\cdot, \cdot)$  in (1), and define  $F : \mathbf{A} \mapsto \mathcal{R}^\alpha$ . Then,

$$F(\mathbf{PAP}^T) = \{\mathbf{\Gamma P}^T | \mathbf{\Gamma} \in \mathcal{R}^\alpha\},$$

where  $\mathbf{P}$  is any permutation matrix.

*Remark.* The permutation equivariance property of sublevel sets is a direct result of the permutation invariance in the objective function. Please refer to Appendix A for a detailed proof.

In the next subsection, by modeling wireless networks as graphs, we show that the permutation equivariance property is universal in radio resource management problems.

### B. Wireless Network as a Graph

A wireless network can be modeled as a directed graph with node and edge features. Naturally, we treat each agent, e.g., a mobile user or a base station, in wireless networks as a node in the graph. An edge is drawn from node  $i$  to node  $j$  if there is a direct communication or interference link with node  $i$  as the transmitter and node  $j$  as the receiver. The node feature incorporates the properties of the agent, e.g., users' weights in the weighted sum rate maximization problem [18]. The edge feature includes the properties of the corresponding channel, e.g., a scalar (or matrix) to denote the channel state of a single-antenna (or multi-antenna) system. We call these graphs generated by the wireless network topology as *wireless channel graphs*. Formally, a wireless channel graph is an ordered tuple  $(V, E, s, t)$ , where  $V$  is the set of nodes,  $E$  is the set of edges,  $s : V \rightarrow \mathbb{C}^{d_1}$  maps a node to its feature, and  $t : E \rightarrow \mathbb{C}^{d_2}$  maps an edge to its feature. Denote  $V = \{v_1, v_2, \dots, v_{|V|}\}$ . Also define the node feature array as  $x \in \mathbb{C}^{1 \times |V| \times d_1}$  with  $x_{1,i,:} = s(v_i)$ , and the adjacency feature array  $A \in \mathbb{C}^{|V| \times |V| \times d_2}$  as

$$A_{i,j,:} = \begin{cases} \mathbf{0}, & \text{if } \{i, j\} \notin E \\ t(\{i, j\}) & \text{otherwise.} \end{cases} \quad (2)$$

where  $\mathbf{0}$  is a zero vector in  $\mathbb{C}^{d_2}$ .

We assign each node  $v_i \in V$  an optimization variable  $\gamma_i \in \mathbb{C}^n$ . Let  $\mathbf{\Gamma} = [\gamma_1, \dots, \gamma_{|V|}]$ , then an optimization problem defined on a wireless channel graph can be written as

$$\mathcal{P} : \underset{\mathbf{\Gamma}}{\text{minimize}} \quad g(\mathbf{\Gamma}, w, A) \quad \text{subject to} \quad Q(\mathbf{\Gamma}, w, A) \leq 0, \quad (3)$$

where  $g(\cdot, \cdot, \cdot)$  denotes the objective function and  $Q(\cdot, \cdot, \cdot)$  denotes the constraint.

Next we elaborate the properties of the radio resource management problems on the wireless channel graphs. Without node features or edge features, a wireless channel graph is a directed graph. As a result, the properties of wireless channel graphs follow the properties of directed graphs. We elaborate the permutation equivariance property of problems on wireless channel graphs next. We call the three dimensions of  $A$  as row, column, and depth. The permutation operator for  $w$  and  $A$  is defined as follows. The left permutation operator  $\Pi_l(\cdot)$  rearranges the rows and the right permutation operator  $\Pi_r(\cdot)$  rearranges columns according to a permutation  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ . Similar to optimization problems on directed graphs, the ones defined on wireless channel graphs have the permutation invariance property. As a result, the sub-level sets of  $g(\cdot, \cdot, \cdot)$  in (3) also have the permutation equivariance property, which is stated below.

**Proposition II.3.** (*Permutation equivariance*) Let  $\mathcal{R}^\alpha$  denote the sublevel set of  $g(\cdot, \cdot, \cdot)$  in (3), and define  $F : (w, A) \mapsto \mathcal{R}^\alpha$ . Then,

$$F(\Pi_r(w), \Pi_r(\Pi_l(A))) = \{\mathbf{\Gamma} \mathbf{P}^T | \mathbf{\Gamma} \in \mathcal{R}^\alpha\},$$

where the permutation matrix  $\mathbf{P}$ , left permutation operator  $\Pi_l(\cdot)$ , and right permutation operator  $\Pi_r(\cdot)$  are associated with the same permutation  $\pi(\cdot)$ .

*Remark.* This result establishes a general permutation equivariance property for radio resource management problems. Proposition II.3 is reduced to the results in [14] if  $w$  is a constant array and  $A \in \mathbb{R}^{|V| \times |V|}$ . Compared with [14], Proposition II.3 is able to handle heterogeneous agents (e.g., users with different resource constraints) and more general channels (e.g., multi-antenna channels) as the heterogeneity can be modeled as node features and multi-antenna channel states can be modeled as edge features. The proof is the same as Proposition II.2 by simply changing notations.



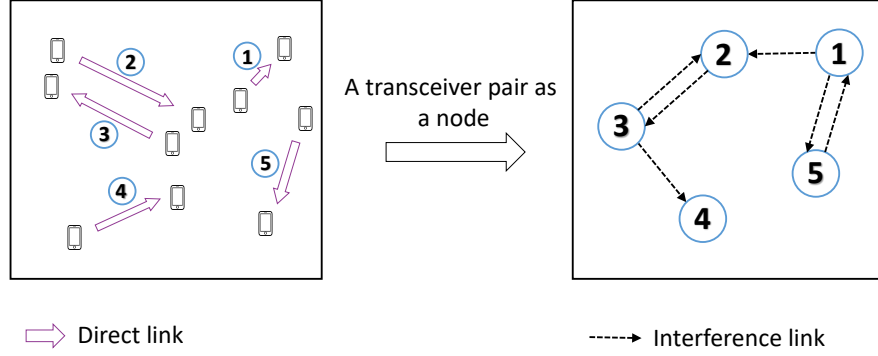


Fig. 1. An illustration of graph modeling of a  $K$ -user interference channel.

### C. Graph Modeling of $K$ -user Interference Channels

In this subsection, as a specific example, we present graph modeling of a classic radio resource management problem, i.e., beamforming for weighted sum rate maximization in a  $K$ -user interference channel. It will be used as the main test setting for the theoretical study in Section IV-C and simulations in Section V. There are in total  $K$  transceiver pairs where each transmitter is equipped with  $N_t$  antennas and each receiver is equipped with a single antenna. Let  $\mathbf{v}_k$  denote the beamformer of the  $k$ -th transmitter. The received signal at receiver  $k$  is  $\mathbf{y}_k = \mathbf{h}_{k,k}^H \mathbf{v}_k s_k + \sum_{j \neq k}^K \mathbf{h}_{j,k}^H \mathbf{v}_j s_j + n_k$ , where  $\mathbf{h}_{j,k} \in \mathbb{C}^{N_t}$  denotes the channel state from transmitter  $j$  to receiver  $k$  and  $n_k \in \mathbb{C}$  denotes the additive noise following the complex Gaussian distribution  $\mathcal{CN}(0, \sigma_k^2)$ .

The signal-to-interference-plus-noise ratio (SINR) for receiver  $k$  is given by

$$\text{SINR}_k = \frac{|\mathbf{h}_{k,k}^H \mathbf{v}_k|^2}{\sum_{j \neq k}^K |\mathbf{h}_{j,k}^H \mathbf{v}_j|^2 + \sigma_k^2}.$$

Denote  $\mathbf{V} = [\mathbf{v}_k]_{k \leq K}$  as the beamforming matrix. The objective is to find the optimal beamformer to maximize the weighted sum rate, and the problem is formulated as

$$\begin{aligned} & \underset{\mathbf{V}}{\text{maximize}} && \sum_{k=1}^K w_k \log_2 (1 + \text{SINR}_k) \\ & \text{subject to} && \|\mathbf{v}_k\|_2^2 \leq P_{\max}, \forall k, \end{aligned} \tag{4}$$

where  $w_k$  is the weight for the  $k$ -th pair.

a) *Graph Modeling*: We view the  $k$ -th transceiver pair as the  $k$ -th node in the graph. As distant agents cause little interference, we draw a directed edge from node  $j$  to node  $k$  only if the distance between transmitter  $j$  and receiver  $k$  is below a certain threshold  $D$ . An illustration of such a graph modeling is shown in Fig. 1. The node feature array  $x \in \mathbb{C}^{1 \times |V| \times (N_t+2)}$  is given by

$$x_{1,k,:} = [w_k, \sigma_k, \mathbf{h}_{k,k}],$$

and the adjacency feature array  $A \in \mathbb{C}^{|V| \times |V| \times N_t}$  is given by

$$A_{j,k,:} = \begin{cases} \mathbf{0}, & \text{if } \{j, k\} \notin E \\ \mathbf{h}_{j,k} & \text{otherwise,} \end{cases}$$

where  $\mathbf{0} \in \mathbb{C}^{N_t}$  is a zero vector. Problem (4) has the permutation equivariance property with respect to  $\mathbf{V}$ ,  $x$ , and  $A$ . To solve this problem efficiently and effectively, the adopted neural network should exploit the permutation equivariance property, and incorporate both node features and edge features. We shall develop an effective neural network architecture to achieve this goal in the next section.

### III. NEURAL NETWORK ARCHITECTURE DESIGN FOR RADIO RESOURCE MANAGEMENT

In this section, we endeavor to develop a scalable neural network architecture for radio resource management problems. A favorable class of GNNs, named, *message passing graph neural networks*, will be identified. The key properties and effective implementation will also be discussed.

#### A. Optimizing Wireless Networks via Graph Neural Networks

Most of existing works on “learning to optimize” approaches to solve problems in wireless networks adopted MLPs as the neural network architecture [7], [9], [11]. Although MLPs can approximate well-behaved functions [24], they suffer from poor performance in data efficiency, robustness, and generalization. A long-standing idea for improving the performance and generalization is to incorporate the structures of the target task into the neural network architecture. In this way, there is no need for the neural network to learn such structures from data, which leads to a more efficient training, and better generalization empirically [21], [25] and provably [22].

As discussed above, the structures of radio resource management problems can be formulated as optimization problems on wireless channel graphs, which enjoy the permutation equivariance property. In machine learning, there are two classes of neural networks that are able to exploit the permutation equivariance property, i.e., graph neural networks (GNNs) [26] and Deep Sets [27]. Compared with Deep Sets, GNNs not only respect the permutation equivariance property but can also model the interactions among the agents. In wireless networks, the agents interact with each other through channels. Thus, GNNs are more favorable than Deep Sets in wireless networks. This motivates us to adopt GNNs to solve radio resource management problems.

### B. Message Passing Graph Neural Networks

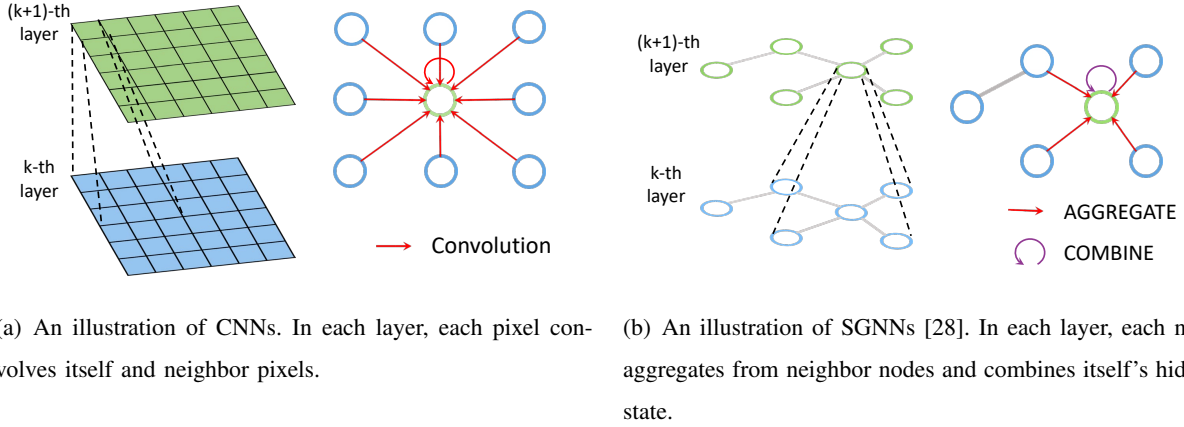


Fig. 2. Illustrations of CNN and SGNNs. CNN can be viewed as a special SGNNs on the grid graph.

In this subsection, we shall identify a favorable class of GNNs for radio resource management problems, which extend CNNs to wireless channel graphs. In traditional machine learning tasks, the data can typically be embedded in a Euclidean space, e.g., images. Recently, there is an increasing number of applications generated from the non-Euclidean spaces that can be naturally modeled as graphs, e.g., point cloud [29] and combinatorial problems [30]. This motivates researchers to develop GNNs [26], which effectively exploit the graph structure. GNNs generalize traditional CNNs, recurrent neural networks, and auto-encoders to the graph tasks. In wireless networks, while the agents are located in the Euclidean space, channel states cannot be embedded in a Euclidean space. Thus, the data in radio resource management problems is also non-Euclidean and neural networks operating on non-Euclidean space are necessary when adopting “learning to optimize” approaches in wireless networks.

As a background, we first introduce CNNs, which operate on Euclidean data. Compared with MLPs, CNNs have shown superior performance in image processing tasks. The motivation for CNNs is that adjacent pixels are meaningful to be considered together in images [23]. Like MLPs, CNNs have a layer-wise structure. In each layer, a 2D convolution is applied to the input. Here we consider a simple CNN with a rectified linear unit and without pooling. In the  $k$ -th layer, for a pixel located at  $(i, j)$ , the update is

$$\mathbf{x}_{(i,j)}^{(k)} = \text{RELU} \left( \sum_{(k,l) \in \mathcal{N}(i,j)} \mathbf{W}_{(k-i,j-l)}^{(k)} \mathbf{x}_{(k,l)}^{(k-1)} \right), \quad (5)$$

where  $\mathbf{x}_{(i,j)}^{(0)}$  denotes pixel  $(i, j)$  of the input image,  $\mathbf{x}_{(i,j)}^{(k)}$  denotes the hidden state of pixel  $(i, j)$  at the  $k$ -th layer,  $\text{RELU}(x) = \text{MAX}(0, x)$  and  $\mathbf{W}_{(\cdot,\cdot)}^{(k)}$  denotes the weight matrix in the  $k$ -th layer, and  $\mathcal{N}(i, j)$  denotes the neighbor pixels of pixel  $(i, j)$ . Specifically, for a convolution kernel of size  $N \times N$ , we have

$$\mathcal{N}(i, j) = \left\{ (k, l) : |k - i| \leq \frac{N-1}{2}, |l - j| \leq \frac{N-1}{2} \right\},$$

and a common choice of  $N$  is 3.

Despite the great success of CNNs in computer vision, they cannot be applied to non-Euclidean data. In [28], CNNs are extended to graphs from a spatial perspective, which is as efficient as CNNs, while enjoying performance guarantees on graph isomorphism test. We refer to this architecture as the *spatial graph convolutional networks* (SGNNs). In each layer of a CNN (5), each pixel aggregates information from neighbor pixels and then updates its state. As an analogy, in each layer of a SGNN, each node updates its representation by aggregating features from its neighbor nodes. Specifically, the update rule of the  $k$ -th layer at vertex  $i$  in a SGNN is

$$\text{SGNNs : } \mathbf{x}_i^{(k)} = \alpha^{(k)} \left( \mathbf{x}_i^{(k-1)}, \phi^{(k)} \left( \left\{ \mathbf{x}_j^{(k-1)} : j \in \mathcal{N}(i) \right\} \right) \right), \quad (6)$$

where  $\mathbf{x}_i^{(0)}$  denotes the input feature of node  $i$ ,  $\mathbf{x}_i^{(k)}$  denotes the hidden state of node  $i$  at the  $k$ -th layer,  $\mathcal{N}(i)$  denotes the set of the neighbors of  $i$ ,  $\phi^{(k)}(\cdot)$  is a set function that aggregates information from the node's neighbors, and  $\alpha^{(k)}(\cdot)$  is a function that combines aggregated information with its own information. An illustration of the extension from CNNs to SGNNs is shown in Fig. 2. Particularly, SGNNs include spatial deep learning for wireless scheduling [10] as a special case.

Despite the success of SGNNs in graph problems, it is difficult to directly apply SGNNs on radio resource allocation problems as they cannot exploit the edge features. This means that

they cannot incorporate channel states in wireless networks. We modify the definition in (6) to exploit edge features and will refer to it as *message passing graph neural networks* (MPGNNs). The update rule for the  $k$ -th layer at vertex  $i$  in an MPGNN is

$$\text{MPGNNs : } \mathbf{x}_i^{(k)} = \alpha^{(k)} \left( \mathbf{x}_i^{(k-1)}, \phi^{(k)} \left( \left\{ \left[ \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right] : j \in \mathcal{N}(i) \right\} \right) \right), \quad (7)$$

where  $\mathbf{e}_{j,i}$  denotes the edge feature of the edge  $(j, i)$  (i.e.,  $A_{j,i}$  in (2)). The output of a  $T$ -layer MPGNN is  $[\mathbf{x}_1^{(T)}, \dots, \mathbf{x}_{|V|}^{(T)}]$ .

The extension from SGNNs to MPGNNs is simple but crucial, due to the following two reasons. First, MPGNNs respect the permutation equivariance property in Proposition II.3. Second, MPGNNs enjoy theoretical guarantees in radio resource management problems (as discussed in Section IV). These two properties are unique for MPGNNs and are not enjoyed by SGNNs.

### C. Key Properties of MPGNNs

MPGNNs enjoy properties that are favorable to solving large-scale radio resource management problems, as discussed in the sequel.

*a) Permutation equivariance:* We first show that MPGNNs satisfy the permutation equivariance property, which leads to easier training and better generalization.

**Proposition III.1.** (*Permutation equivariance in MPGNNs*) *Viewing the input output mapping of MPGNNs defined in (7) as  $\Phi : (w, A) \mapsto \Gamma$ , we have*

$$\Phi(\Pi_r(w), \Pi_r(\Pi_l(A))) = \Phi(w, A) \mathbf{P}^T$$

*for any permutation matrix  $\mathbf{P}$ .*

*Remark.* Please refer to Appendix B for a detailed proof.

*b) Ability to generalize to different problem scales:* In MLPs, the input or output size must be the same during training and testing. Hence, the problem size in the test dataset must be equal or less than the problem scale in the training dataset [7]. This means that MLP based methods cannot be directly applied to a different problem size. In MPGNNs, each node has a copy of two sub neural networks, i.e.,  $\alpha^{(k)}(\cdot)$  and  $\phi^{(k)}(\cdot)$ , whose input-output dimensions are invariant with the problem scale. Thus, we can train MPGNNs on small-scale problems and apply them to large-scale problems.

c) *Fewer training samples*: The required number of training samples for MPGNNs is much smaller than that for MLPs. The first reason is training sample reusing. For each training sample, each node receives a permuted version of it and processes it with  $\alpha^{(k)}(\cdot)$  and  $\phi^{(k)}(\cdot)$ . Thus, each training sample is reused  $L$  times for training  $\{\alpha^{(k)}\}$  and  $\{\phi^{(k)}\}$ , where  $L$  is the problem scale. Second, input and output dimensions of the aggregation and combination functions in MPGNNs are much smaller than the original problem, which allows the use of much fewer parameters in neural networks.

d) *High computational efficiency*: In each layer, an aggregation function is applied to all the edges and a combination function is applied to all the nodes. Thus, the time complexity for each layer is  $\mathcal{O}(|E|+|V|)$  and the overall time complexity for an  $L$ -layer MPGNN is  $\mathcal{O}(L(|E|+|V|))$ . The time complexity grows *linearly* with the number of agents when the maximal degree of the graph is bounded. Note that in MPGNNs, the aggregation function and combination function on each node can be executed in parallel. When the MPGNNs are fully parallelized, e.g., on powerful GPUs, the time complexity is  $\mathcal{O}(LD)$ , where  $D$  is the maximal degree of the graph. This is a *constant* time complexity when the maximal degree of the graph is bounded. We will verify this observation via simulations in Fig. 4.

#### D. An Effective Implementation of MPGNNs

In this subsection, we propose an effective implementation of MPGNNs for radio resource management problems, named, the *wireless channel graph convolution network* (WCGCN), which is able to incorporate both agent-related features and channel-related features. The design space for MPGNNs (7) is to choose the set aggregation function  $\phi(\cdot)$  and the combination function  $\alpha(\cdot)$ .

As general set functions are difficult to implement, an efficient implementation of  $\phi(\cdot)$  was proposed in [31], which has the following form

$$\phi(\{x_1, \dots, x_n\}) = \psi(\{h(x_1), \dots, h(x_n)\}),$$

where  $x_i, 1 \leq i \leq n$  are the elements in the set,  $\psi(\cdot)$  is a simple function, e.g., max or sum, and  $h(\cdot)$  is some existing neural network architecture, e.g., linear mappings or MLPs. For  $\alpha(\cdot)$  and  $h(\cdot)$ , linear mapping is adopted in popular GNN architectures (e.g., GCN [32] and S2V [33]). Nevertheless, as discussed in Section IV in [13], linear mappings have difficulty handling continuous features, which is ubiquitous in wireless networks (e.g., CSI). We adopt MLPs as

$\alpha(\cdot)$  and  $h(\cdot)$  for their approximation ability [24]. MLP processing unit enables WCGCN to exploit complicated agent-related features and channel-related features in wireless networks.

For the aggregation function  $\psi(\cdot)$ , we notice that the following property holds if we use  $\psi(\cdot) = \text{MAX}(\cdot)$ .

**Theorem III.1.** (*Robustness to feature corruptions*) [34] Suppose  $\mathbf{u} : \mathcal{X} \rightarrow \mathbb{R}^p$  such that  $\mathbf{u} = \text{MAX}_{x_i \in S}$  and  $f = \gamma \circ \mathbf{u}$ . Then,

$$(a) \forall S, \exists \mathcal{C}_S, \mathcal{N}_S \in \mathcal{X}, f(T) = f(S) \text{ if } \mathcal{C}_S \subset T \subset \mathcal{N}_S;$$

$$(b) |\mathcal{C}_S| \leq p$$

Theorem III.1 states that  $f(S)$  remains the same up to corruptions of the input if all the features in  $\mathcal{C}_S$  are preserved and  $\mathcal{C}_S$  only contains a limited number of features, which is smaller than  $p$ . By specifying it to problems in wireless networks, the output of a layer remains unchanged even when the CSI is heavily corrupted on some links. In other words, it is robust to missing CSI.

We next specify the architecture for the WCGCN, which aligns with traditional optimization algorithms. First, in traditional optimization algorithms, each iteration outputs an updated version of the optimization variables. In the WCGCN, each layer outputs an updated version of the optimization variables. Second, these algorithms are often time-invariant systems, e.g., gradient descent, WMMSE [18], and FPlinQ [35]. Thus, we share weights among different layers of the WCGCN, and the updates are

$$\begin{aligned} \mathbf{y}_i^{(k)} &= \text{MLP2} \left( \mathbf{x}_i^{(k-1)}, \text{MAX}_{j \in \mathcal{N}(i)} \{ \text{MLP1} \left( \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \} \right), \\ \mathbf{x}_i^{(k)} &= \sigma(\mathbf{y}_i^{(k)}), \end{aligned} \tag{8}$$

where MLP1 and MLP2 are two different MLPs, and  $\sigma$  is a normalization function that depends on applications. For example, for the power control problem, we constrain the power between 0 and 1, and  $\sigma$  can be a sigmoid function, i.e.,  $\sigma(x) = \frac{1}{1+\exp(-x)}$ .

Besides the benign properties of MPGNNs, WCGCN enjoys several desirable properties for solving large-scale radio resource management problems. First, the WCGCN can effectively exploit features in multi-antenna systems with heterogeneous agents (e.g., channel states in multi-antenna systems and users' weights in weighted sum rate maximization). This is because WCGCN adopts MLP as processing units instead of linear mappings. This enables it to solve a wider class of radio resource management tasks than existing works [10], [13], [14] (e.g.,

beamforming problems and weighted sum rate maximization). Second, it is robust to partial and imperfect CSI as suggested in Theorem III.1.

#### IV. THEORETICAL ANALYSIS OF MPGNN-BASED RADIO RESOURCE MANAGEMENT

In this section, we investigate performance and generalization of MPGNNs. We first prove the equivalence between MPGNNs and a class of distributed algorithms, which include many classic algorithms for radio resource management as special examples, e.g., WMMSE [18] and FPlinQ [35]. Based on this observation, we analyze the performance of MPGNN-based methods for weighted sum rate maximization problem.

##### A. Simplifications

To provide theoretical guarantees for “learning to optimize” approaches for solving radio resource management problems, it is critical to understand the performance and generalization of neural network-based methods. Unfortunately, the training and generalization of neural networks are still open problems. We make several commonly adopted simplifications to make the performance analysis tractable. First, we focus on the MPGNN class instead of any specific neural network architecture such as GCNs. Following Lemma 5 and Corollary 6 in [28], we can design an MPGNN with MLP processing units as powerful as the MPGNN class, and thus this simplification well serves our purpose. Second, we target at proving the existence of an MPGNN with performance guarantee. Because we train the neural network with a stochastic gradient descent with limited training samples during the simulations, we may not find the corresponding neural network parameters. While this may leave some gap between the theory and practice, our result is an important first step. These two simplifications have been commonly adopted in the performance analysis of GNNs [28], [36], [37].

##### B. Equivalence of MPGNNs and Distributed Optimization

Compared with the neural network-based radio resource management, optimization-based radio resource management has been well studied. Thus, it is desirable to make connections between these two types of methods. In [36], the equivalence between some special types of GNNs and graph optimization algorithms was proved. Inspired by this result, we shall establish the equivalence between MPGNNs and a class of distributed radio resource management algorithms.



We first give a brief introduction to *distributed local algorithms*, following [38]. The maximal degree of the nodes in the graph is assumed to be bounded. Distributed local algorithms are a class of iterative algorithms in a multi-agent system. In each iteration, each agent sends messages to its neighbors, receives messages from its neighbors, and updates its state based on the received messages. The algorithm terminates after a constant number of iterations.

We focus on a sub-class of distributed local algorithms, titled, *multiset broadcasting distributed local algorithms* (MB-DLA) [38], which include a wide range of radio resource management algorithms in wireless networks, e.g., DTP [39], WMMSE [18], and FPLinQ [35]. Multiset and broadcasting refer to the way for receiving and sending messages, respectively. Denote  $\mathbf{x}_i^{(l)}$  as the state of node  $i$  at the  $l$ -th iteration, and the MB-DLA is shown in Algorithm 1.

---

**Algorithm 1** Multiset broadcasting distributed local algorithm [38]

---

- 1: Initialize all internal states  $\mathbf{x}_k^{(0)}, \forall k$ .
  - 2: **for** communication round  $t = 1, \dots, T$  **do**
  - 3:   agent  $k$  sends  $h_1^{(t)}(\mathbf{x}_k^{(t-1)})$  to all its edges,  $\forall k$
  - 4:   agent  $k$  receives  $\left\{ \mathbf{m}_{j,k}^{(t)} \mid \mathbf{m}_{j,k}^{(t)} = h_2^{(t)} \left( h_1^{(t)}(\mathbf{x}_k^{(t-1)}), \mathbf{e}_{j,k} \right), j \in \mathcal{N}(k) \right\}$  from the edges,  $\forall k$
  - 5:   agent  $k$  updates its internal state  $\mathbf{x}_k^{(t)} = g_1^{(t)} \left( \mathbf{x}_k^{(t-1)}, g_2^{(t)} \left( \left\{ \mathbf{m}_{j,k}^{(t)} : j \in \mathcal{N}(k) \right\} \right) \right), \forall k$ .
  - 6: **end for**
  - 7: Output  $\mathbf{x}_k^{(T)}$ .
- 

The equivalence between MPGNNs and MB-DLAs roots in the similarity in their definitions. In each iteration of an MB-DLA, each agent aggregates messages from neighbor agents and updates its local state. In each layer of an MPGNN, each node aggregates features from neighbor nodes. The equivalence can be drawn if we view the agents as nodes in a graph and messages as the features. The following proposition states the equivalence of MPGNNs and MB-DLAs formally.

**Theorem IV.1.** *Let  $\text{MB-DLA}(T)$  denote the class of MB-DLA with  $T$  iterations and  $\text{MPGNN}(T)$  as the class of MPGNNs with  $T$  layers, then the following two conclusions hold.*

- 1) *For any  $\text{MPGNN}(T)$ , there exists a distributed local algorithm in  $\text{MB-DLA}(T)$  that solves the same set of problems as  $\text{MPGNN}(T)$ .*
- 2) *For any algorithm in  $\text{MB-DLA}(T)$ , there exists an  $\text{MPGNN}(T)$  that solves the same set of problems as this algorithm.*

*Remark.* Please refer to Appendix C for a detailed proof.

The equivalence allows us to analyze the performance of MPGNNs by studying the performance of MB-DLAs. The first result shows that MPGNNs are *at most* as powerful as MB-DLAs. The implication is that if we can prove that there is no MB-DLA capable of solving a specific radio resource management problem, then MPGNNs cannot solve it. This can be used to prove a performance upper bound of MPGNNs. The second result shows that MPGNNs are *as powerful as* MB-DLAs in radio resource management problems. This implies that if we are able to identify an MB-DLA that solves a radio resource management problem well, then there exists an MPGNN performs better or at least competitive. The generalization is also as good as the corresponding MB-DLA. We shall give a specific example on sum rate maximization in the next subsection.

### C. Performance and Generalization of MPGNNs

In this subsection, we use the tools developed in the last subsection to analyze the performance and generalization of MPGNNs in the sum rate maximization problem. The analysis is built on the observation that a classic algorithm for the sum rate maximization problem, i.e., WMMSE, is an MB-DLA under some conditions, which is formally stated below. We shall refer to the MB-DLA corresponding to WMMSE as WMMSE-DLA.

**Proposition IV.1.** *When the maximal number of interference neighbors is bounded by some constant, then WMMSE with a constant number of iterations is an MB-DLA.*

*Remark.* When the problem sizes in the training dataset and test dataset are the same, we can always assume that the number of interference neighbors is a common constant. The restriction of a constant number of interference neighbors only influences the generalization. Please refer to Appendix D for a detailed proof.

*a) Performance:* Proposition IV.1 shows that WMMSE is an MB-DLA. Thus, when the problem sizes in the training dataset and test dataset are the same, there exists an MPGNN whose performance is as good as WMMSE. As the WMMSE is hand-crafted, it is not optimal in terms of the number of iterations. By employing a unsupervised loss function, we expect that MPGNNs can learn an algorithm which has fewer iterations and may possibly enjoy better performance. In Fig. 3, we observe that a 1-layer MPGNN outperforms WMMSE with 10 iterations and a 2-layer MPGNN outperforms WMMSE with 30 iterations.

*b) Generalization:* To avoid the excessive training cost, it is desirable to first train a neural network on small-scale problems and then generalize it to large-scale ones. An intriguing question is when such generalization is reliable. Compared with WMMSE, WMMSE-DLA has two constraints: Both the number of iterations and the maximal number of interference neighbors should be bounded by some constants. As agents that are far away cause little interference, the number of interference neighbors can be assumed to be fixed when the user density is kept the same. As a result, compared with WMMSE with a fixed number of iterations, the performance of MPGNNs is stable when the user density in the test dataset is the user density in the training dataset multiplied by a constant. We will verify this by simulations in Table IV and Table VII.

## V. SIMULATION RESULTS

In this section, we provide simulation results to verify the effectiveness of the proposed neural network architecture for three applications. The first application is sum rate maximization in a Gaussian interference channel, which is a classic application for deep learning-based methods. We use this application to compare the proposed method with MLP-based methods [9] and optimization-based methods [18]. The second application is weighted sum rate maximization, and the third application is beamformer design. The last two problems cannot be solved by existing methods in [10], [13], [14].

For the neural network setting, we adopt a 3-layer WGCN, implemented by Pytorch Geometric [31]. We apply unsupervised training without labeled samples, and the loss function is defined as

$$\ell = -\mathbb{E} \left( \sum_{k=1}^K w_k \log_2 \left( 1 + \frac{|\mathbf{h}_{k,k}^H \mathbf{v}_k|^2}{\sum_{j \neq k}^K |\mathbf{h}_{j,k}^H \mathbf{v}_j|^2 + \sigma_k^2} \right) \right)$$

where the expectation is taken over all the channel realizations. To optimize the neural network, we adopt the adam optimizer with a learning rate 0.001.

### A. Sum Rate Maximization

We first consider the sum rate maximization problem in a single-antenna Gaussian interference channel. This problem is a special case of (4) with  $N_t = 1$ ,  $h_{j,k} \sim \mathcal{CN}(0, 1)$ ,  $\forall j, k$ , and  $w_k = 1, \forall k$ .

We consider the following benchmarks for comparison.

- **WMMSE** [18]: This is a classic optimization-based algorithm for sum utility maximization in MIMO interfering broadcast channels. We run WMMSE for 100 iterations.
- **Strongest**: We find a fixed proportion of pairs with the largest channel gain  $|h_{i,i}|$ , and set the power of these pairs as  $P_{\max}$  while the power levels for remaining pairs are set to 0. This is a simple baseline algorithm without any knowledge of interference links.
- **PCNet** [9]: PCNet is an MLP based method particularly designed for the sum rate maximization problem with single-antenna channels.

We use  $10^4$  training samples for WCGCN and  $10^7$  training samples for PCNet. For a specific parameter setting of WCGCN (8), we set the hidden units of MLP1 in (8) as  $\{5, 32, 32\}$ , MLP2 as  $\{35, 16, 1\}$ , and  $\sigma(\cdot)$  as sigmoid function.<sup>2</sup> The performance of different methods is shown in Table I. The SNR and number of users are kept the same in the training and test dataset. For all the tables shown in this section, the entries are (weighted) the sum rates achieved by different methods normalized by the sum rate of WMMSE. We see that both PCNet and WCGCN achieve near-optimal performance when the problem scale is small. As the problem scale becomes large, the performance of PCNet approaches Strongest. This shows that it can hardly learn any valuable information about interference links. Nevertheless, the performance of WCGCN is stable as the problem size increases. Thus, GNNs are more favorable than MLPs for medium-scale or large-scale problems.

TABLE I  
AVERAGE SUM RATE UNDER EACH SETTING. THE RESULTS ARE NORMALIZED BY THE SUM RATE OF WMMSE.

	$K = 10, 0\text{dB}$	$K = 30, 0\text{dB}$	$K = 50, 0\text{dB}$	$K = 10, 10\text{dB}$	$K = 30, 10\text{dB}$	$K = 50, 10\text{dB}$
WCGCN	100.0%	97.9%	97.1%	103.1%	103.4%	102.5%
PCNet	98.9%	87.4%	79.7%	101.8%	74.0%	67.0%
Strongest	87.1%	82.8%	80.6%	74.4%	70.0%	68.9%

We further compare the performance of WCGCN and WMMSE with different numbers of iterations. We use the system setting  $K = 50$ , SNR= 10dB and the results are shown in Fig. 3. From the figure, we see that a 1-layer WCGCN outperforms WMMSE with 10 iterations and a 2-layer WCGCN outperforms WMMSE with 30 iterations. This indicates that by adopting the

<sup>2</sup>The performance of WCGCN is not sensitive to the number of hidden units.

unsupervised loss function, WCGCN can learn a much better message-passing algorithm than the handcrafted WMMSE.

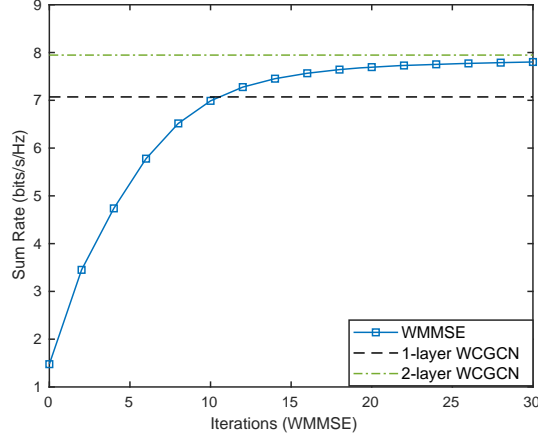


Fig. 3. A comparison between WCGCN and WMMSE with different numbers of iterations.

### B. Weighted Sum Rate Maximization

In this application, we consider  $K$  single-antenna transceiver pairs within a  $A \times A$  area. The transmitters are randomly located in the  $A \times A$  area while each receiver is uniformly distributed within  $[d_{\min}, d_{\max}]$  from the corresponding transmitter. We adopt the channel model from [19] and use 10000 training samples for each setting. To reduce the CSI training overhead, we assume  $h_{j,k}$  is available to WCGCN only if the distance between transmitter  $j$  and receiver  $k$  is within 500 meters. To provide a performance upper bound, global CSI is assumed to be available to WMMSE. The weights for weighted sum rate maximization, i.e.,  $w_k$  in (4), are generated from a uniform distribution in  $[0, 1]$  in both training and test dataset. For a specific parameter setting of WCGCN (8), we set the hidden units of MLP1 as  $\{5, 32, 32\}$ , MLP2 as  $\{35, 16, 1\}$ , and  $\sigma(\cdot)$  as sigmoid function.

*a) Performance comparison:* We first test the performance of WCGCN when the number of pairs is the same in the training and test dataset. Specifically, we consider  $K = 50$  pairs in a  $1000\text{m} \times 1000\text{m}$  region. We test the performance of WCGCN with different values of  $d_{\min}$  and  $d_{\max}$ , as shown in Table II. The entries in the table are the sum rates achieved by different methods. We observe that WCGCN with local CSI achieves competitive performance to WMMSE with global CSI.

TABLE II  
AVERAGE SUM RATE PERFORMANCE OF 50 TRANSCIVER PAIRS.

$(d_{\min}, d_{\max})$	(2m,65m)	(10m,50m)	(30m,70m)	(30m,30m)
WCGCN	97.8%	97.5%	96.5%	96.8%

Next, to test the generalization capability of the proposed method, we train WCGCN on a wireless network with tens of users and test it on wireless networks with hundreds or thousands of users, as shown in the following two simulations.

*b) Generalization to larger scales:* We first train the WCGCN with 50 pairs in a  $1000\text{m} \times 1000\text{m}$  region. We then change the number of pairs in the test set while the density of users (i.e.,  $A^2/K$ ) is fixed. The results are shown in Table III. It can be observed that the performance is stable as the number of users increases. It also shows that WCGCN can well generalize to larger problem scales, which is consistent with our analysis.

TABLE III  
GENERALIZATION TO LARGER PROBLEM SCALES BUT SAME DENSITY.

Number of Links		$K = 200$	$K = 400$	$K = 600$	$K = 800$	$K = 1000$
Field length (m)		2000	2828	3464	4000	4772
$(d_{\min}, d_{\max})$	(10m,50m)	98.3%	98.9%	98.8%	98.9%	98.9%
	(30m,30m)	98.1%	98.2%	98.7%	98.6%	98.7%

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

Number of Links		$K = 100$	$K = 200$	$K = 300$	$K = 400$	$K = 500$
Field length		1000	1000	1000	1000	1000
$(d_{\min}, d_{\max})$	(10m,50m)	97.6% (+0.1%)	97.0% (−0.5%)	95.9% (−1.6%)	95.6% (−1.9%)	95.3% (−2.2%)
	(30m,30m)	96.4% (−0.1%)	96.0% (−0.5%)	94.9% (−1.6%)	94.5% (−2.0%)	94.5% (−2.0%)

*c) Generalization to higher densities:* In this test, we first train the WCGCN with 50 pairs in a  $1000\text{m} \times 1000\text{m}$  region. We then change the number of pairs in the test set while fixing the area size. The results are shown in Table IV and the performance loss compared with  $K = 50$

is shown in the bracket. The performance is stable up to a 4-fold increase in the density, and good performance is achieved even when there is a 10-fold increase in the density.

### C. Beamformer Design

In this subsection, we consider the beamforming for sum rate maximization in (4). Specifically, we consider  $K$  transceiver pairs within a  $A \times A$  area, where the transmitters are equipped with multiple antennas and each receiver is equipped with a single antenna. The transmitters are generated uniformly in the area and the receivers are generated uniformly within  $[d_{\min}, d_{\max}]$  from the corresponding transmitters. We adopt the channel model in [19] and use 50000 training samples for each setting. The assumption of the available CSI for WCGCN and WMMSE is the same as the previous subsection. In WCGCN, a complex number is treated as two real numbers. For a specific parameter setting of WCGCN (8), we set the hidden units of MLP1 as  $\{6N_t, 64, 64\}$ , MLP2 as  $\{64 + 4N_t, 32, 2N_t\}$ , and  $\sigma(\mathbf{x}) = \frac{\mathbf{x}}{\max(\|\mathbf{x}\|_2, 1)}$ .

*a) Performance comparison:* We first test the performance of WCGCN when the number of pairs in the training dataset and the number of pairs in the test dataset are the same. Specifically, we consider  $K = 50$  pairs in a 1000 meters by 1000 meters region and each transmitter is equipped with 2 antennas. We test the performance of WCGCN with different  $d_{\min}$  and  $d_{\max}$ . The results are shown in Table V. We observe that WCGCN achieves comparable performance to WMMSE with local CSI, demonstrating the applicability of the proposed method to multi-antenna systems.

TABLE V

AVERAGE SUM RATE PERFORMANCE OF 50 TRANSCIVER PAIRS WITH  $N_t = 2$ . THE RESULTS ARE NORMALIZED BY THE SUM RATE OF WMMSE.

$(d_{\min}, d_{\max})$	(2m,65m)	(10m,50m)	(30m,70m)	(30m,30m)
WCGCN	97.1%	96.0%	94.1%	96.2%

*b) Generalization to larger scales:* We first train the WCGCN with 50 pairs in a 1000 meters by 1000 meters region with  $N_t = 2$ . We then change the number of pairs while the density of users (i.e.,  $A^2/K$ ) is fixed. The results are shown in Table VI. The performance is stable as the number of users increases, which is consistent with our theoretical analysis.

TABLE VI  
GENERALIZATION TO LARGER PROBLEM SCALES BUT SAME DENSITY.

Number of Links		$K = 200$	$K = 400$	$K = 600$	$K = 800$	$K = 1000$
Field length		2000	2828	3464	4000	4772
$(d_{\min}, d_{\max})$	(2m,65m)	97.3%	97.3%	97.2%	97.2%	97.2%
	(10m,50m)	96.8%	96.7%	96.5%	96.5%	96.4%

*c) Generalization to larger densities:* We first train the WCGCN with 50 pairs on a 1000 meters by 1000 meters region with  $N_t = 2$ . We then change the number of pairs while fix the area size. The results are shown in Table VII and the performance loss is shown in the bracket. The performance is stable up to a 2-fold increase in the density and satisfactory performance is achieved up to a 4-fold increase in the density. The performance deteriorates when the density grows, which indicates that extra training is needed when the density in the test dataset is much larger than that of the training dataset.

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

Number of Links		$K = 100$	$K = 200$	$K = 300$	$K = 400$	$K = 500$
Field length		1000	1000	1000	1000	1000
$(d_{\min}, d_{\max})$	(2m,65m)	97.0% (−0.1%)	95.8%(−1.3%)	94.5% (−2.6%)	92.5% (−4.6%)	91.4% (−5.7%)
	(10m,50m)	95.7% (−0.3%)	94.4% (−1.6%)	93.0% (−3.0%)	92.0% (−4.0%)	90.7%(−5.3%)

*d) Computation time comparison:* This test compares the running time of different methods for different problem scales. We run “WCGCN GPU” on GeForce GTX 1080Ti while the other methods on Intel(R) Xeon(R) CPU E5-2643 v4 @ 3.40GHz. The implementation of neural networks exploits the parallel computation of GPU while WMMSE is not able to do so due to its sequential computation flows. The running time is averaged over 50 problem instances and shown in Fig. 4. The speedup compared with WMMSE becomes large as the problem scale increases. This benefits from the low computational complexity of WCGCN. As shown in the figure, the computational complexity of WCGCN CPU is linear and WCGCN GPU is nearly a constant, which is consistent with our analysis in Section III-C. Remarkably, WCGCN is able to solve the problem with 1000 users within 6 milliseconds.



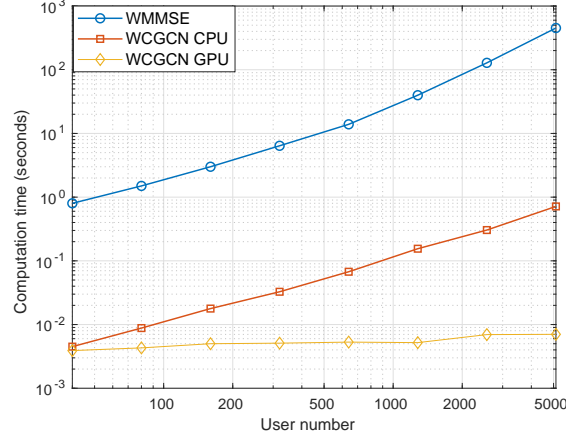


Fig. 4. Computation time comparison of different methods.

## VI. CONCLUSIONS

In this paper, we developed a scalable neural network architecture based on GNNs to solve radio resource management problems. In contrast to existing learning based methods, we focused on the neural architecture design to meet the key performance requirements, including low training cost, high computational efficiency, and good generalization. Moreover, we theoretically connected learning based methods and optimization based methods, which casts light on the performance guarantee of learning to optimize approaches. We believe that this investigation will lead to profound implications in both theoretical and practical aspects. As for future directions, it will be interesting to investigate the distributed deployment of MPGNNs for radio resource management in wireless networks, and extend our theoretical results to more general application scenarios.

## APPENDIX A

### PROOF OF PROPOSITION II.2

Following Proposition II.1, we have

$$g(\Gamma, \mathbf{A}) = g(\Gamma \mathbf{P}^T, \mathbf{P} \mathbf{A} \mathbf{P}^T), \quad Q(\Gamma, \mathbf{A}) = Q(\Gamma \mathbf{P}^T, \mathbf{P} \mathbf{A} \mathbf{P}^T) \quad (9)$$

for any variable  $\Gamma$ , adjacency matrix  $\mathbf{A}$ , and permutation matrix  $\mathbf{P}$ .

For any  $\Gamma \in \mathcal{R}^\alpha$ , we have

$$g(\Gamma \mathbf{P}^T, \mathbf{P} \mathbf{A} \mathbf{P}^T) = g(\Gamma, \mathbf{A}) \leq \alpha, \quad Q(\Gamma \mathbf{P}^T, \mathbf{P} \mathbf{A} \mathbf{P}^T) = Q(\Gamma, \mathbf{A}) \leq 0. \quad (10)$$

Combining (9) and (10), we have

$$\begin{aligned}
F(\mathbf{PAP}^T) &= \{\Gamma | g(\Gamma, \mathbf{PAP}^T) \leq \alpha, Q(\Gamma, \mathbf{PAP}^T) \leq 0\} \\
&= \{\Gamma \mathbf{P}^T | g(\Gamma \mathbf{P}^T, \mathbf{PAP}^T) \leq \alpha, Q(\Gamma \mathbf{P}^T, \mathbf{PAP}^T) \leq 0\} = \{\Gamma \mathbf{P}^T | g(\Gamma, \mathbf{A}) \leq \alpha, Q(\Gamma, \mathbf{A}) \leq 0\} \\
&= \{\Gamma \mathbf{P}^T | \Gamma \in \mathcal{R}^\alpha\}.
\end{aligned}$$

## APPENDIX B

### PROOF OF PROPOSITION III.1

In the original graph, denote the input feature of node  $i$  as  $\mathbf{x}_i^{(0)}$ , the edge feature of edge  $(j, i)$  as  $\mathbf{e}_{j,i}$ , and the output of the  $k$ -th layer of node  $i$  as  $\mathbf{x}_i^{(k)}$ . In the permuted graph, denote the input feature of node  $i$  as  $\hat{\mathbf{x}}_i^{(0)}$ , the edge feature of edge  $(j, i)$  as  $\hat{\mathbf{e}}_{j,i}$ , and the output of the  $k$ -th layer for node  $i$  as  $\hat{\mathbf{x}}_i^{(k)}$ . Due to the permutation relationship, we have

$$\hat{\mathbf{e}}_{\pi(j), \pi(i)} = \mathbf{e}_{j,i}, \quad \hat{\mathbf{x}}_{\pi(i)}^{(0)} = \mathbf{x}_i^{(0)}, \quad \mathcal{N}(\pi(i)) = \{\pi(j), j \in \mathcal{N}(i)\}. \quad (11)$$

We prove the result by induction. First, we have  $\hat{\mathbf{x}}_{\pi(i)}^{(0)} = \mathbf{x}_i^{(0)}$  as in (11). Assume  $\hat{\mathbf{x}}_{\pi(i)}^{(k)} = \mathbf{x}_i^{(k)}$ . In the  $k+1$ -th layer, the following update rule is applied

$$\begin{aligned}
\mathbf{x}_i^{(k+1)} &= \alpha^{(k+1)} \left( \mathbf{x}_i^{(k)}, \phi^{(k+1)} \left\{ \left[ \mathbf{x}_j^{(k)}, \mathbf{e}_{j,i} \right] : j \in \mathcal{N}(i) \right\} \right), \\
\hat{\mathbf{x}}_{\pi(i)}^{(k+1)} &= \alpha^{(k+1)} \left( \hat{\mathbf{x}}_{\pi(i)}^{(k)}, \phi^{(k+1)} \left\{ \left[ \hat{\mathbf{x}}_j^{(k)}, \hat{\mathbf{e}}_{j, \pi(i)} \right] : j \in \mathcal{N}(\pi(i)) \right\} \right).
\end{aligned} \quad (12)$$

Following (11), (12) and the induction hypothesis, we have  $\mathbf{x}_i^{(k+1)} = \hat{\mathbf{x}}_{\pi(i)}^{(k+1)}$ . As a result,  $\mathbf{x}_i^{(T)} = \hat{\mathbf{x}}_{\pi(i)}^{(T)}$  and thus  $\Phi(\Pi_r(w), \Pi_r(\Pi_l(A))) = \Phi(w, A) \mathbf{P}^T$ .

## APPENDIX C

### PROOF OF THEOREM IV.1

In MB-DLAs, the maximal degree of nodes should be bounded by some constant, denoted by  $\Delta$ . The update of MB-DLA at the  $l$ -th iteration can be written as

$$\text{MB-DLA : } \quad \hat{\mathbf{m}}_{i,j}^{(l)} = h \left( h_1(\hat{\mathbf{x}}_i^{(l-1)}), \mathbf{e}_{i,j} \right), \quad \hat{\mathbf{x}}_i^{(l)} = g_1^{(l)} \left( \hat{\mathbf{x}}_i^{(l-1)}, g_2^{(l)} \left( \left\{ \hat{\mathbf{m}}_{j,i}^{(l)} : j \in \mathcal{N}(i) \right\} \right) \right). \quad (13)$$

The update of an MPGNN at the  $k$ -layer can be written as

$$\text{MPGNNs : } \quad \mathbf{x}_i^{(k)} = \alpha^{(k)} \left( \mathbf{x}_i^{(k-1)}, \phi^{(k)} \left\{ \left[ \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right] : j \in \mathcal{N}(i) \right\} \right). \quad (14)$$

1) We first show that the inference stage of an MPGNN can be viewed as an MB-DLA, which is proved by induction. Before the algorithm and neural network start, both  $\hat{\mathbf{x}}_i^{(0)}$  and  $\mathbf{x}_i^{(0)}$  are

node features and thus  $\hat{\mathbf{x}}_i^{(0)} = \mathbf{x}_i^{(0)}$ . We assume  $\hat{\mathbf{x}}_i^{(t-1)} = \mathbf{x}_i^{(t-1)}$ . At the  $t$ -th iteration, the message  $\hat{\mathbf{m}}_{i,j}^{(t)} = [\mathbf{x}_j^{(t-1)}, \mathbf{e}_{j,i}]$  is passed from agent  $j$  to agent  $i$ . Then agent  $i$  updates its local state as  $g_1^{(t)} = \alpha^{(t)}$  and  $g_2^{(t)} = \phi^{(t)}$ . By doing so, we have  $\hat{\mathbf{x}}_i^{(t)} = \mathbf{x}_i^{(t)}$ .

2) We show that (13) can be written in the form of (14). Before the algorithm and neural network start, both  $\hat{\mathbf{x}}_i^{(0)}$  and  $\mathbf{x}_i^{(0)}$  are node features and thus  $\mathbf{x}_i^{(0)} = \hat{\mathbf{x}}_i^{(0)}$ . We assume  $\mathbf{x}_i^{(t-1)} = \hat{\mathbf{x}}_i^{(t-1)}$ . Let  $\mathbf{y}_j = [\mathbf{x}_j^{(t-1)}, \mathbf{e}_{j,i}]$ . At the  $t$ -th iteration, node  $i$  aggregates features from neighbor nodes that form a multiset  $S(i) = \{\mathbf{y}_j : j \in \mathcal{N}(i)\}$ . We order the elements in  $S(i)$  according to their first coordinates. Let  $\tau_i(S), 1 \leq i \leq \Delta$  denote the function that selects the  $i$ -th element in a multiset  $S$ ,  $\chi_1 : \mathbf{x} \mapsto \mathbf{x}_{1:d_1}$ ,  $\chi_2 : \mathbf{x} \mapsto \mathbf{x}_{d_1+1:d_2}$ . Taking

$$\phi^{(t)}(S(i)) = g_2^{(t)}(\{h(h_1(\chi_1(\tau_1(S(i))))), \chi_2(\tau_1(S(i))), \dots, h(h_1(\chi_1(\tau_\Delta(S(i))))), \chi_2(\tau_\Delta(S(i)))\}),$$

and  $\alpha^{(t)} = g_1^{(t)}$ , we then obtain  $\mathbf{x}_i^{(t)} = \hat{\mathbf{x}}_i^{(t)}$ . This completes the proof.

## APPENDIX D

### PROOF OF PROPOSITION IV.1

WMMSE [18] is a classic algorithm for weighted sum rate maximization in MIMO interfering broadcast channels. The WMMSE algorithm considers a  $K$  cell interfering broadcast channel where base station (BS)  $k$  serves  $I_k$  users. Denote  $\mathbf{H}_{i_k,j}$  as the channel from base station  $j$  to user  $i_k$ ,  $\mathbf{V}_{i_k}$  as the beamformer that BS  $k$  uses to transmit symbols to user  $i_k$ ,  $w_{i_k}$  as the weight of user  $i_k$ , and  $\sigma_{i_k}^2$  as the variance of noise for user  $i_k$ . The problem formulation is

$$\begin{aligned} & \underset{\mathbf{V}}{\text{maximize}} && \sum_{k=1}^K \sum_{i=1}^{I_k} w_{i_k} \log \det \left( \mathbf{I} + \mathbf{H}_{i_k,k} \mathbf{V}_{i_k} \mathbf{V}_{i_k}^H \mathbf{H}_{i_k,k}^H \left( \sum_{(l,j) \neq (i,k)} \mathbf{H}_{i_k,j} \mathbf{V}_j \mathbf{V}_j^H \mathbf{H}_{i_k,j}^H + \sigma_{i_k}^2 \mathbf{I} \right)^{-1} \right) \\ & \text{subject to} && \text{Tr}(\mathbf{V}_{i_k} \mathbf{V}_{i_k}^H) \leq P_{\max}, \forall k, \end{aligned}$$

The WMMSE algorithm is shown in Algorithm 2.

We first model this system as a graph. We treat the  $i_k$ -th user as the  $i_k$ -th node in the graph. The node features are  $[w_{i_k}, \sigma_{i_k}, \mathbf{H}_{i_k,k}]$ . The internal state of node  $i_k$  at the  $p-1$ -th iteration is  $[\mathbf{U}_{i_k}^{(p-1)}, \mathbf{W}_{i_k}^{(p-1)}, \mathbf{V}_{i_k}^{(p-1)}, w_{i_k}, \sigma_{i_k}, \mathbf{H}_{i_k,k}]$ . An edge is drawn from the  $l_j$ -th node to the  $i_k$ -th node if there is an interference link between the  $j$ -th BS and the  $i_k$ -th user. The edge feature of the edge  $(l_j, i_k)$  is  $\mathbf{e}_{l_j, i_k} = [\mathbf{H}_{i_k,j}, \mathbf{H}_{k,l_j}]$ .

We show that a WMMSE algorithm with  $T$  iterations is an MB-DLA with *at most*  $2T$  iterations. We update the variables  $\mathbf{U}_{i_k}$  and  $\mathbf{W}_{i_k}$  at the odd iterations while updating the variable  $\mathbf{V}_{i_k}$  at the even iterations. Specifically, at the  $p$ -th iteration with  $p$  being an odd number, the  $l_j$ -th node broadcasts its state  $\mathbf{V}_{l_j}^{(p-1)}$  along its edges. The edge  $(l_j, i_k)$  processes the message by forming

$\mathbf{m}_{l_j, i_k} = \mathbf{H}_{i_k, j} \mathbf{V}_{l_j}^{(p-1)} \mathbf{V}_{l_j}^{(p-1)H} \mathbf{H}_{i_k, j}^H$  and the node  $i_k$  receives the message set  $\{\mathbf{m}_{l_j, i_k}, \forall l, j\}$ . The agent  $i_k$  first sums over the messages  $\mathbf{M}_{i_k} = \sum_{l, j} \mathbf{m}_{l_j, i_k}$ . Then the  $i_k$ -th node updates its internal state as  $\mathbf{U}_{i_k}^{(p)} = (\mathbf{M}_{i_k} + \sigma_{i_k} \mathbf{I})^{-1} \mathbf{H}_{i_k, k} \mathbf{V}_{i_k}^{(p-1)}$  and  $\mathbf{W}_{i_k}^{(p)} = \left( \mathbf{I} - \mathbf{U}_{i_k}^{(p)H} \mathbf{H}_{i_k, k} \mathbf{V}_{i_k}^{(p-1)} \right)^{-1}$ . Thus, the update at the odd iterations can be written in the form of (13) as follows

$$\begin{aligned} \mathbf{x}_{l_j}^{(p-1)} &= [\mathbf{U}_{l_j}^{(p-1)}, \mathbf{W}_{l_j}^{(p-1)}, \mathbf{V}_{l_j}^{(p-1)}, w_{l_j}, \sigma_{i_k}, \mathbf{H}_{l_j, j}], \quad \mathbf{e}_{l_j, i_k} = [\mathbf{H}_{i_k, j}, \mathbf{H}_{k, l_j}], \\ \mathbf{m}_{l_j, i_k} &= h^{(p)} \left( h_1 \left( \mathbf{x}_{i_k}^{(p-1)} \right), \mathbf{e}_{l_j, i_k} \right) = \mathbf{H}_{i_k, j} \mathbf{V}_{l_j}^{(p-1)} \mathbf{V}_{l_j}^{(p-1)H} \mathbf{H}_{i_k, j}^H, \\ \mathbf{M}_{i_k} &= g_2^{(p)}(\{\mathbf{m}_{l_j, i_k}, \forall l, j\}) = \sum_{l, j} \mathbf{m}_{l_j, i_k}, \\ \mathbf{U}_{i_k}^{(p)} &= (\mathbf{M}_{i_k} + \sigma_{i_k} \mathbf{I})^{-1} \mathbf{H}_{i_k, k} \mathbf{V}_{i_k}^{(p-1)}, \quad \mathbf{W}_{i_k}^{(p)} = \left( \mathbf{I} - \mathbf{U}_{i_k}^{(p)H} \mathbf{H}_{i_k, k} \mathbf{V}_{i_k}^{(p-1)} \right)^{-1}, \\ g_1^{(p)}(\mathbf{x}_{i_k}^{(p-1)}, \mathbf{M}_{i_k}) &= [\mathbf{U}_{i_k}^{(p)}, \mathbf{W}_{i_k}^{(p)}, \mathbf{V}_{i_k}^{(p-1)}, w_{i_k}, \sigma_{i_k}, \mathbf{H}_{i_k, k}]. \end{aligned}$$

At the  $p$ -th iteration where  $p$  is even, the  $l_j$ -th node broadcasts its state  $[\mathbf{V}_{l_j}^{(p-1)}, \mathbf{W}_{l_j}^{(p-1)}]$  along its edges. The edge  $(l_j, i_k)$  processes the message by forming  $\mathbf{m}_{l_j, i_k} = \mathbf{H}_{l_j, k}^H \mathbf{U}_{l_j}^{(p-1)} \mathbf{W}_{l_j}^{(p-1)} \mathbf{U}_{l_j}^{(p-1)H} \mathbf{H}_{l_j, k}$ . Node  $i_k$  receives the message set  $\{\mathbf{m}_{l_j, i_k}, \forall l, j\}$ . The agent  $i_k$  first sums over the messages  $\mathbf{M}_{i_k} = \sum_{l, j} \mathbf{m}_{l_j, i_k}$ . Then the  $i_k$ -th node updates its internal state as  $\mathbf{V}_{i_k}^{(p)} = w_{i_k} (\mathbf{M}_{i_k} + \mu_{i_k}^* \mathbf{I})^{-1} \mathbf{H}_{i_k, k}^H \mathbf{U}_{i_k}^{(p-1)} \mathbf{W}_{i_k}^{(p-1)}$ . Thus, the update during the even iterations can be written in the form of (13), i.e.,

$$\begin{aligned} \mathbf{x}_{l_j}^{(p-1)} &= [\mathbf{U}_{l_j}^{(p-1)}, \mathbf{W}_{l_j}^{(p-1)}, \mathbf{V}_{l_j}^{(p-1)}, w_{l_j}, \sigma_{i_k}, \mathbf{H}_{l_j, j}], \quad \mathbf{e}_{l_j, i_k} = [\mathbf{H}_{i_k, j}, \mathbf{H}_{k, l_j}], \\ \mathbf{m}_{l_j, i_k} &= h^{(p-1)} \left( h_1 \left( \mathbf{x}_{i_k}^{(p-1)} \right), \mathbf{e}_{l_j, i_k} \right) = w_{l_j} \mathbf{H}_{l_j, k}^H \mathbf{U}_{l_j}^{(p-1)} \mathbf{W}_{l_j}^{(p-1)} \mathbf{U}_{l_j}^{(p-1)H} \mathbf{H}_{l_j, k}, \\ \mathbf{M}_{i_k} &= g_2^{(p)}(\{\mathbf{m}_{l_j, i_k}, \forall l, j\}) = \sum_{l, j} \mathbf{m}_{l_j, i_k}, \\ \hat{\mathbf{V}}_{i_k}(\mu_{i_k}) &= w_{i_k} (\mathbf{M}_{i_k} + \mu_{i_k} \mathbf{I})^{-1} \mathbf{H}_{i_k, k}^H \mathbf{U}_{i_k}^{(p-1)} \mathbf{W}_{i_k}^{(p-1)}, \quad \mu_{i_k}^* = \underset{\mu_{i_k} \geq 0, \text{Tr}(\hat{\mathbf{V}}_{i_k}^H \hat{\mathbf{V}}_{i_k}) \leq P_{\max}}{\text{argmin}} \mu_{i_k}, \\ \mathbf{V}_{i_k}^{(p)} &= w_{i_k} (\mathbf{M}_{i_k} + \mu_{i_k}^* \mathbf{I})^{-1} \mathbf{H}_{i_k, k}^H \mathbf{U}_{i_k}^{(p-1)} \mathbf{W}_{i_k}^{(p-1)}, \\ g_1^{(p)}(\mathbf{x}_{i_k}^{(p-1)}, \mathbf{M}_{i_k}) &= [\mathbf{U}_{i_k}^{(p-1)}, \mathbf{W}_{i_k}^{(p-1)}, \mathbf{V}_{i_k}^{(p)}, w_{i_k}, \sigma_{i_k}, \mathbf{H}_{i_k, k}]. \end{aligned}$$

This completes the proof for Proposition IV.1.

## REFERENCES

- [1] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "A graph neural network approach for scalable wireless power control," in *Proc. IEEE Global Commun. Conf. (GLOBECOM'19) Workshops*, Waikoloa, HI, USA, Dec. 2019.
- [2] M. Chiang, P. Hande, and T. Lan, "Power control in wireless cellular networks," *Found. Trends Networking*, vol. 2, no. 4, pp. 381–533, 2008.
- [3] E. Björnson, M. Bengtsson, and B. Ottersten, "Optimal multiuser transmit beamforming: A difficult problem with a simple solution structure," *IEEE Signal Process. Mag.*, vol. 31, no. 4, pp. 142–148, 2014.

---

**Algorithm 2** WMMSE Algorithm [18]

---

```

1: Initialize  $V_{i_k}$ .
2: for  $t = 0 \dots T$  do
3:    $U_{i_k} \leftarrow \left( \sum_{(l,j)} H_{i_k,j} V_{l_j} V_{l_j}^H H_{i_k,j}^H + \sigma_{i_k}^2 I \right)^{-1} H_{i_k,k} V_{i_k}$ 
4:    $W_{i_k} \leftarrow \left( I - U_{i_k}^H H_{i_k,k} V_{i_k} \right)^{-1}$ 
5:    $V_{i_k} \leftarrow w_{i_k} \left( \sum_{l,j} w_{l_j} H_{l_j,k}^H U_{l_j} W_{l_j} U_{l_j}^H H_{l_j,k} + \mu_{i_k}^* I \right)^{-1} H_{i_k,k}^H U_{i_k} W_{i_k}$ 
6: end for
7: Output  $V_{i_k}$ .

```

---

- [4] Y. Shi, J. Zhang, B. O'Donoghue, and K. B. Letaief, "Large-scale convex optimization for dense wireless cooperative networks," *IEEE Trans. Signal Process.*, vol. 63, pp. 4729–4743, Sept. 2015.
- [5] Y. Shi, J. Zhang, K. B. Letaief, B. Bai, and W. Chen, "Large-scale convex optimization for ultra-dense cloud-RAN," *IEEE Wireless Commun.*, vol. 22, pp. 84–91, Jun. 2015.
- [6] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016.
- [7] 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.
- [8] W. Lee, M. Kim, and D. Cho, "Deep power control: Transmit power control scheme based on convolutional neural network," *IEEE Commun. Lett.*, vol. 22, pp. 1276–1279, Apr. 2018.
- [9] F. Liang, C. Shen, W. Yu, and F. Wu, "Towards optimal power control via ensembling deep neural networks," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1760–1776, 2020.
- [10] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE J. Sel. Areas Commun.*, vol. 37, Jun. 2019.
- [11] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "LORM: Learning to optimize for resource management in wireless networks with few training samples," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 665–679, 2020.
- [12] W. Xia, G. Zheng, Y. Zhu, J. Zhang, J. Wang, and A. P. Petropulu, "A deep learning framework for optimization of MISO downlink beamforming," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1866–1880, 2020.
- [13] M. Lee, G. Yu, and G. Y. Li, "Graph embedding based wireless link scheduling with few training samples," *arXiv preprint arXiv:1906.02871*, 2019.
- [14] M. Eisen and A. Ribeiro, "Optimal wireless resource allocation with random edge graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 2977–2991, 2020.
- [15] J. Dong, J. Zhang, Y. Shi, and J. H. Wang, "Faster activity and data detection in massive random access: A multi-armed bandit approach," *arXiv preprint arXiv:2001.10237*, 2020.
- [16] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *arXiv preprint arXiv:1811.06128*, 2018.
- [17] L. Liang, H. Ye, G. Yu, and G. Y. Li, "Deep-learning-based wireless resource allocation with application to vehicular networks," *Proc. IEEE*, 2019.
- [18] Q. Shi, M. Razaviyayn, Z. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a mimo interfering broadcast channel," *IEEE Trans. Signal Process.*, vol. 59, pp. 4331–4340, Sept. 2011.
- [19] Y. Shi, J. Zhang, and K. B. Letaief, "Group sparse beamforming for green cloud-RAN," *IEEE Trans. Wireless Commun.*, vol. 13, pp. 2809–2823, May 2014.

- [20] Y. Shi, J. Zhang, W. Chen, and K. B. Letaief, "Enhanced group sparse beamforming for green cloud-RAN: A random matrix approach," *IEEE Trans. Wireless Commun.*, vol. 17, pp. 2511–2524, Apr. 2018.
- [21] C. Sun, J. Wu, and C. Yang, "Data representation for deep learning with priori knowledge of symmetric wireless tasks," *arXiv preprint arXiv:2005.08510*, 2020.
- [22] K. Xu, J. Li, M. Zhang, S. Du, K. Kawarabayashi, and S. Jegelka, "What can neural networks reason about?," in *Proc. Int. Conf. Learning Representations*, Apr. 2020.
- [23] W. Brendel and M. Bethge, "Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet," *Proc. Int. Conf. Learning Representation*, May 2019.
- [24] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [25] S. Ravanbakhsh, J. Schneider, and B. Póczos, "Equivariance through parameter-sharing," in *Proc. Int. Conf. Mach. Learning*, pp. 2892–2901, Jul. 2017.
- [26] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Networks Learning Syst.*, pp. 1–21, 2020.
- [27] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. Adv. Neural Inform. Process. Syst.*, pp. 3391–3401, Dec. 2017.
- [28] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," *Proc. Int. Conf. Learning Representation*, May 2019.
- [29] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graphics*, vol. 38, no. 5, pp. 1–12, 2019.
- [30] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Proc. Adv. Neural Inform. Process. Syst.*, pp. 539–548, Dec. 2018.
- [31] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," in *Proc. Int. Conf. Learning Representations Workshops*, May 2019.
- [32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *Proc. Int. Conf. Learning Representation*, Apr. 2017.
- [33] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," in *Proc. Int. Conf. Mach. Learning*, pp. 2702–2711, Jun. 2016.
- [34] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, pp. 652–660, Honolulu, HI, USA, Jul. 2017.
- [35] K. Shen and W. Yu, "FPLinQ: A cooperative spectrum sharing strategy for device-to-device communications," in *Proc. IEEE Int. Symp. Inform. Theory*, pp. 2323–2327, Jun. 2017.
- [36] R. Sato, M. Yamada, and H. Kashima, "Approximation ratios of graph neural networks for combinatorial problems," in *Proc. Adv. Neural Inform. Process. Syst.*, pp. 4083–4092, Dec. 2019.
- [37] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. Reutter, and J. P. Silva, "The logical expressiveness of graph neural networks," in *Proc. Int. Conf. Learning Representations*, Apr. 2020.
- [38] L. Hella, M. Järvisalo, A. Kuusisto, J. Laurinharju, T. Lempinen, K. Luosto, J. Suomela, and J. Virtama, "Weak models of distributed computing, with connections to modal logic," *Distributed Comput.*, vol. 28, no. 1, pp. 31–53, 2015.
- [39] M. Kubisch, H. Karl, A. Wolisz, L. C. Zhong, and J. Rabaey, "Distributed algorithms for transmission power control in wireless sensor networks," in *Proc. IEEE Wireless Commun. Networking*, pp. 558–563, New Orleans, LA, USA, Mar. 2003.