Optimal Wireless Resource Allocation With Random Edge Graph Neural Networks

Mark Eisen [ID] , *Member, IEEE*, and Alejandro Ribeiro [ID]

Abstract—We consider the problem of optimally allocating resources across a set of transmitters and receivers in a wireless network. The resulting optimization problem takes the form of constrained statistical learning, in which solutions can be found in a model-free manner by parameterizing the resource allocation policy. Convolutional neural networks architectures are an attractive option for parameterization, as their dimensionality is small and does not scale with network size. We introduce the random edge graph neural network (REGNN), which performs convolutions over random graphs formed by the fading interference patterns in the wireless network. The REGNN-based allocation policies are shown to retain an important permutation equivariance property that makes them amenable to transference to different networks. We further present an unsupervised model-free primal-dual learning algorithm to train the weights of the REGNN. Through numerical simulations, we demonstrate the strong performance REGNNs obtain relative to heuristic benchmarks and their transference capabilities.

Index Terms—Power allocation, deep learning, graph neural networks, interference channel.

I. INTRODUCTION

WIRELESS systems are integral to large scale intelligent systems, from robotics to the Internet of Things (IoT). The design of such systems requires optimal balancing of the numerous utilities and constraints that define the operating point of large networks of wireless connected devices. At a high level, such optimal design problems can be viewed as the allocation of a finite set of resources to achieve strong average performance over the randomly varying wireless channel. While these optimization problems can be easily formulated, they tend to be intractable as they are most often non-convex and infinite dimensional [1]. Some simplification is attained by working in the Lagrangian dual domain [1], [2] and subsequently using dual descent methods [3]–[5], or, alternatively, with heuristic optimization and scheduling methods [6]–[9].

All such approaches invariably require accurate system models and may require prohibitively large computational cost. As

emergent applications demand growth in scale and complexity, modern machine learning and statistical regression techniques have been explored as alternatives to solve wireless resource allocation problems. Machine learning methods train a learning model, such as a neural network (NN), to approximate the behavior of resource allocation strategies for a wide variety of problems. A common approach is to exploit supervised learning techniques to train a NN that approximates the behavior of an existing heuristic to reduce computational cost during execution [10]–[13]. The use of supervised learning, however, is limited by the availability of heuristics and hindered by their suboptimality. Supervised learning methods require these solutions to build training sets and may meet heuristic performance but never exceed it.

Perhaps the more compelling justification for statistical learning solutions to wireless resource allocation problems lies in their reliance on *data* over *models*. That is, by treating the resource allocation problem itself as a form of statistical regression, we obtain a means of training policies that solve the optimal allocation problem directly rather than via a training set [14]–[19]. This *unsupervised* approach exceeds the capabilities of supervised learning in that it can be applied to any arbitrary resource allocation problem and has the potential to exceed performance of existing heuristics. In addition, it is possible to make unsupervised learning *model-free* by relying on interactions with the wireless system. We probe with a candidate resource allocation policy, observe its outcome, and use this information to discover a better policy [17].

There nonetheless remains the practical challenge of training models that can meet the scale of modern wireless systems. *Fully connected* neural networks (FCNNs) may seem appealing due to their well known universal approximation property [10], [17]. However, FCNNs are also well known to be unworkable except in small scale problems. Scalability is attained in the processing of signals in time and space with *convolutional* neural networks (CNNs). Recognizing this fact has led to proposals that adapt CNNs to wireless resource allocation problems [12], [13], [16]. A particularly enticing alternative is the use of a spatial CNN that exploits the spatial geometry of wireless networks to attain scalability to large scale systems with hundreds of nodes [20].

In this paper we develop a different alternative to scalability that leverages graph neural networks (GNNs) [21], [22]. GNNs are neural network architectures replacing the convolutional filter banks of CNNs with graph convolutional filter banks defined as polynomials on a matrix representation of a graph [22]. We propose here a variation which we call a random edge

graph neural network (REGNN). REGNNs take as inputs the state of communication links and the state of the nodes of the network to produce resource allocation functions through the composition of layers which are themselves the composition of graph convolutional filter banks with pointwise nonlinearities. Through a combination of design choices as well as theoretical and numerical analyses this paper demonstrates that REGNNs have the following three properties:

Scalability: REGNNs are defined by a number of parameters that is chosen independent of the number of nodes in the network. This enables training in large scale systems. We demonstrate in numerical experiments the possibility to scale to networks with several hundred nodes.

Permutation invariance: We prove that if REGNN parameters are optimal for a certain network, they are optimal for all of its permutations. This allows transference across different networks for as long as they are not far from permutations of each other.

Transference: A given REGNN can be executed in any graph independent of shape and size. In particular, this makes it possible to train and execute in different networks. We demonstrate in numerical experiments the ability to transfer a REGNN across families of networks as well as the ability to train in networks of moderate size (with a few tens of nodes) and execute with good performance in large scale networks (with hundreds of nodes).

In conjunction with the model-free algorithmic learning approach developed in [17], we obtain a unified framework for learning effective resource allocation policies in large scale wireless systems. We point out related prior work that also considers graph neural network architectures for wireless resource allocation [23]–[25] – see Remark 4.

We begin the paper by introducing a generic formulation of wireless resource allocation problems in which we seek a instantaneous resource allocation policy given a set of random fading states and random node states (Section II). Such a formulation has many applications, ranging from multiple access to wireless control systems (Section II-A). The problem generally cannot be solved exactly, but can be addressed through statistical learning techniques by parameterizing the resource allocation policy. We propose the use of random edge graph neural networks (REGNNs) to parameterize the policy by viewing the random fading links between transceivers as a graph with random edges (Section III). This parameterization is a generalization of the popular convolution neural networks and has low-dimensionality that makes it scalable for large wireless networks.

Any policy of practical use should have the ability to be implemented on varying network toplgies. We present in Theorem 1 a so-called permutation invariance of the REGNN with respect to the underlying graph structure of its inputs (Section IV). We further establish the permutation equivariance of both the optimal unparameterized resource allocation policy (Section IV-A) and the learned REGNN (Section IV-B). We present an unsupervised, model-free primal-dual algorithm to train the REGNN filter tensor weights without requiring explicit model knowledge (Section V). We conclude with a comprehensive set of numerical simulations that demonstrate the strong performance of learned REGNN resource allocation policies (Section VI), including their ability to transfer to varying network topologies (Section VI-B).

II. OPTIMAL RESOURCE ALLOCATION IN WIRELESS COMMUNICATION SYSTEMS

Consider a wireless system made up of a set of m transmitters and n receivers. Each transmitter $i \in \{1, m\}$ is paired with a single receiver $r(i) \in \{1, n\}$. Multiple transmitters may be paired with the same receiver. We denote as $R_j := \{i : r(i) = j\}$ the set of transmitters i paired with receiver j. Fig. 1(a) illustrate ad hoc networks in which $n = m$ and the pairing of transmitters and receivers is bijective. Figs. 1(b) illustrates a cellular uplink in which $m > n$ and the map is surjective with R_j containing the users in the catchment area of base station j. To model a cellular downlink we need to replicate the base station node to produce a bijective map from base stations to nodes. Although not required we have in mind large scale wireless systems with n ranging from several tens to several hundreds.

Time is slotted with connections between agents in a time slot characterized by fading channel coefficients. We use $h_{ii}(t)$ to denote the channel between transmitter i and receiver $r(i)$ and $h_{ij}(t)$ to denote the channel between transmitter i and receiver $r(j)$ at time slot t. All channels are arranged in the matrix $\mathbf{H}(t) \in \mathbb{R}^{m \times m}$ with entries $[\mathbf{H}(t)]_{ij} = h_{ij}(t)$. In addition to channel states there are also separate state variables $x_i(t)$ representing a random state of the communication between i and $r(i)$ – such as, e.g., the number of packets that arrive in the time slot. These node states are collected in the vector $\mathbf{x}(t)$ with entries $[\mathbf{x}(t)]_i = x_i(t)$. Our goal is to map state observations $\mathbf{H}(t)$ and $\mathbf{x}(t)$ to a resource allocation function $\mathbf{p}(t) = \mathbf{p}(\mathbf{H}(t), \mathbf{x}(t))$. Allocating resources $\mathbf{p}(t)$ when the system state pair is $\mathbf{H}(t), \mathbf{x}(t)$ produces a vector reward of the form

$$\mathbf{r}(t) = \mathbf{f}\left(\mathbf{p}(t); \mathbf{H}(t), \mathbf{x}(t)\right), \tag{1}$$

where $\mathbf{f}$ is a function from the joint space of resource allocations and states to the space of rewards. This abstract model encompasses several problems of practical importance as we illustrate in Section II-A.

In fast fading scenarios the instantaneous value of the reward $\mathbf{r}(t)$ in (1) is not directly experienced by end users. Rather, end users experience the long term average across time slots. Assuming stationarity and independence of subsequent state realizations, this time average can be replaced by an expectation. Let then $m(\mathbf{H}, \mathbf{x})$ represent a probability distribution of channel and node states and approximate the long term average reward by its limit which we can equate to the expected reward,

$$\mathbf{r} = \mathbb{E}\left[\mathbf{f}\left(\mathbf{p}\left(\mathbf{H}, \mathbf{x}\right); \mathbf{H}, \mathbf{x}\right)\right] = \int \mathbf{f}\left(\mathbf{p}\left(\mathbf{H}, \mathbf{x}\right); \mathbf{H}, \mathbf{x}\right) dm(\mathbf{H}, \mathbf{x}). \tag{2}$$

The goal of the optimal wireless system design problems we study in this paper is to find the instantaneous resource allocation policy $\mathbf{p}(\mathbf{H}, \mathbf{x})$ that optimizes the expected reward in (2).

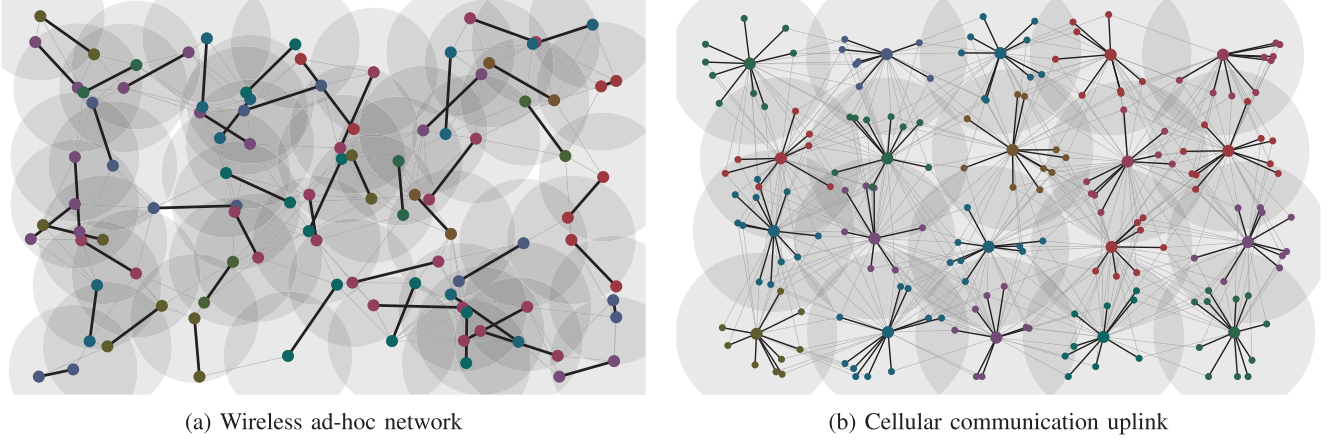(a) Wireless ad-hoc network   (b) Cellular communication uplink

Fig. 1. Resource allocation in large scale wireless networks. Wireless links connect transmitters to receivers (bold lines) but wireless transmission also generates interference to other receivers (thin lines). We formulate optimal resource allocation as machine learning problem over the interference graph that we solve using random edge graph neural networks (REGNNs).

Specifically, introduce a utility function $u_0(\mathbf{r})$ and a set of utility constraints $\mathbf{u}(\mathbf{r}) \leq \mathbf{0}$ to formulate the optimization problem

$$\mathbf{p}^*(\mathbf{H}, \mathbf{x}) = \operatorname*{argmax} \quad u_0(\mathbf{r}),$$
$$\text{s.t. } \mathbf{r} = \mathbb{E}\left[\mathbf{f}\left(\mathbf{p}\left(\mathbf{H}, \mathbf{x}\right); \mathbf{H}, \mathbf{x}\right)\right],$$
$$\mathbf{u}(\mathbf{r}) \geq \mathbf{0}, \quad \mathbf{p}(\mathbf{H}, \mathbf{x}) \in \mathcal{P}(\mathbf{H}, \mathbf{x}). \quad (3)$$

In the problem in (3) we find an average reward $\mathbf{r}$ that maximizes the utility $u_0(\mathbf{r})$ while making sure the utility constraints $\mathbf{u}(\mathbf{r}) \leq \mathbf{0}$ are satisfied. We do so by searching for the resource allocation $\mathbf{p}^*(\mathbf{H}, \mathbf{x})$ that produces such expected reward according to (2). We have also added the constraint $\mathbf{p}(\mathbf{H}, \mathbf{x}) \in \mathcal{P}(\mathbf{H}, \mathbf{x})$ to represent (simple) constraints on allowable resource allocations; see, Section II-A. It is worth stressing that, due to the equivalence between long term and average reward in (2), we are able to solve long term resource allocation problems with the standard statistical optimization problem in (3).

Note that most prior heuristic resource allocation methods maximize an instantaneous reward instead of the average reward considered in (3). Generally speaking, maximizing an instantaneous reward requires the solving of a constrained optimization problem for each instance of the channel and node states $\mathbf{H}$ and $\mathbf{x}$—for complex problems with fast varying channel conditions, it often poses an infeasible computational burden to solve the problem for every channel instance. The formulation in (3) defines a single policy $\mathbf{p}(\mathbf{H}, \mathbf{x})$ that maximizes the ergodic performance of the system; thus, the global optimization need only be solved once offline. At execution, the instantaneous resource allocation is obtained by passing instantaneous states through $\mathbf{p}(\mathbf{H}, \mathbf{x})$ at marginal computational cost [1].

The utilities in (3) are design choices and can be made convex. The ergodic constraint [cf. (2)], however, incorporates the function $\mathbf{f}$ in (1) which is typically not convex. In addition, fading channel realizations are a dense set and we are interested in cases where the number of transmitters is large. This makes solution of (3) intractable and motivates the use of various heuristics; e.g., [6]. If we are considering heuristics in general, we can, in particular, use data driven heuristics where we propose some

resource allocation, observe its outcome, and use this information to update the resource allocation policy. To that end we follow an interpretation of (3) originally developed in [17], in which we identify it as a constrained statistical learning problem. Consider then a parameter $\boldsymbol{\theta} \in \mathbb{R}^q$ and a function family $\boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \boldsymbol{\theta})$ that we use to generate resource allocations

$$\mathbf{p}(\mathbf{H}, \mathbf{x}) = \boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \boldsymbol{\theta}). \quad (4)$$

To fix ideas, say that we choose the family $\boldsymbol{\Phi}$ to consist of quadratic plus linear functions of the form $\boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}, \boldsymbol{\theta}) = (1/2)\boldsymbol{\theta}_1^T \mathbf{H} \boldsymbol{\theta}_1 + \boldsymbol{\theta}_2^T \mathbf{x}$. Or, as done in many recent machine learning applications, we make $\boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}, \boldsymbol{\theta})$ the output of a neural network; e.g., [10], [17], [18]. In any event, with a given parametrization we can substitute (4) into (3) to obtain a problem in which the optimization over resource allocations $\mathbf{p}(\mathbf{H}, \mathbf{x})$ is replaced with an optimization over the set of parameter vectors $\boldsymbol{\theta}$,

$$\boldsymbol{\theta}^* = \operatorname*{argmax} \quad u_0(\mathbf{r}),$$
$$\text{s.t.} \quad \mathbf{r} = \mathbb{E}\left[\mathbf{f}\left(\boldsymbol{\Phi}\left(\mathbf{H}, \mathbf{x}; \boldsymbol{\theta}\right); \mathbf{H}, \mathbf{x}\right)\right],$$
$$\mathbf{u}(\mathbf{r}) \geq \mathbf{0}, \quad \boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \boldsymbol{\theta}) \in \mathcal{P}(\mathbf{H}, \mathbf{x}). \quad (5)$$

Notice that in (5) we include the hard constraint $\boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \boldsymbol{\theta}) \in \mathcal{P}(\mathbf{H}, \mathbf{x})$. We point out that this constraint can often be satisfied in practice through careful design of $\boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \boldsymbol{\theta})$. In particular, the parameterization can be augmented to include an additional output operation that projects the allocation onto $\mathcal{P}(\mathbf{H}, \mathbf{x})$.

The problems in (3) and (5) look similar but are different in three important ways: (i) The original optimization problem in (3) is a functional optimization problem given that, in general, $\mathbf{H}$ and $\mathbf{x}$ belong to dense sets. The parametrized problem in (5) is on the q-dimensional variable $\boldsymbol{\theta}$. (ii) The parametrized problem in (5) can be solved without having access to a model for the function $\mathbf{f}$. It suffices to have the ability to probe the system with a resource allocation $\boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \boldsymbol{\theta})$ and measure the outcome $\mathbf{f}(\boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \boldsymbol{\theta}); \mathbf{H}, \mathbf{x})]$ as we detail in Section V – see also [17]. This is impossible in (3) whose solution requires access to the model $\mathbf{f}$. (iii) The learning parametrization reduces the space of

allowable resource allocations so that the optimal solution of (5) entails a loss of optimality relative to the solution of (3).

The latter point calls for judicious choice of the learning parametrization. E.g., if we use a fully connected neural network in (4) we can rely on universality results to claim a small loss of optimality – along with other interesting theoretical claims [17]. However, fully connected neural networks do not work beyond simple low dimensional problems and our interest is in problems where we have m^2 input variables in $\mathbf{H}$ and m input variables in $\mathbf{x}$ with large m. We propose here to use graph neural networks (Section III) which we will demonstrate provide an scalable parametrization that permits finding good solutions to (5) with large values of m (Section VI). Before introducing GNNs we present some examples of resource allocation problems belonging to the family of abstract problems introduced in (3).

A. Examples

Multiple access AWGN channel: Terminals communicating with associated receivers on a shared channel. A standard instantaneous performance metric of interest here is the capacity experienced by each user under noise and interference. The ith element of $\mathbf{f}(\mathbf{p}(\mathbf{H}, \mathbf{x}; \mathbf{H}, \mathbf{x})$ may then denote the instantaneous capacity achieved by transmitter i. In a channel subject to additive white Gaussian noise (AWGN) and multi-user interference and assuming the use of capacity achieving codes this is given by

$$f_i(\mathbf{p}; \mathbf{H}, \mathbf{x}) := \log\left(1 + \frac{h_{ii}p_i(\mathbf{H}, \mathbf{x})}{1 + \sum_{j \neq i} h_{ji}p_j(\mathbf{H}, \mathbf{x})}\right). \quad (6)$$

Defining the performance as in (6) in the constraint in (3) reflects a maximization with respect to the long term capacity experienced by the users. A constraint of the form $\mathbf{u}(\mathbf{r}) \geq \mathbf{0}$ enforces a minimum average capacity for all users if $\mathbf{u}(\mathbf{r}) := \mathbf{r} - \mathbf{c}_{\min}$ to achieve fairness. Power constraints can be enforced via the set $\mathcal{P} = \{\mathbf{p} : \mathbf{0} \leq \mathbf{p} \leq \mathbf{p}_0\}$ and the utility u_0 can be chosen to be the sum rate $u_0(\mathbf{r}) = \sum_i r_i$ or a proportional fair utility $u_0(\mathbf{r}) = \sum_i \log(r_i)$—see, e.g., [26]. In this problem there is no node state $\mathbf{x}$ that plays a role in the system's design.

Multiple access with user demand: We augment the previous example to incorporate varying traffic demand or information generation at each node. Here, the state x_i reflects the rate of information collection or the data arrival rate at the ith transmitter. An additional performance metric is the average arrival rate $\mathbf{r}' := \mathbb{E}(\mathbf{x})$. A necessary constraint on the system is that average capacity exceeds observed or target average collection rates $\mathbf{r}_0$. We achieve that with the constraint

$$\mathbf{u}(\mathbf{r}') := \mathbf{r}_0 - \mathbf{r}' \geq \mathbf{0}. \quad (7)$$

Another form of user demand comes in the context of fairness, in which the state $\mathbf{x}$ may reflect the achieved lifetime rates of all of the users with ergodic average $\mathbf{y} := \mathbf{E}(\mathbf{x})$. A weighted sum-rate would thus prioritize users who have not been given sufficient access to the channel, i.e.,

$$\mathbf{u}(\mathbf{r}, \mathbf{y}) = \sum_i \frac{r_i}{y_i}. \quad (8)$$

These problems make full use of the generality of (3) by containing both the fading channel states and associated performance metric in the capacity function given in (6), as well as a node state given by either data collection and associated coupled constraint given by (7), or lifetime rate and associated weighted objective given by (8). We point out that the weighted sum-rate problem, in particular, is typically solved by the WMMSE method [6], although such a method would rely on explicit model knowledge and does not generally support additional system constraints.

Random access wireless control systems: A more complex example modeled by (3) concerns resource allocation in a wireless control system. Consider that transmitters are sending plant state information to a shared receiver/base station to compute control inputs over a common random access channel that is subject to potential packet collisions. Given the direct and interference channel states and transmission powers, we define a function $q(p_i, h_{ii}, p_j, h_{ij}) \to [0, 1]$ that gives a probability of collision between transmitters i and j. We are interested in the probability of successful transmission of transmitter i, i.e.,

$$q_i(\mathbf{p}; \mathbf{H}, \mathbf{x}) := \prod_{j \neq i} (1 - q(p_i, h_{ii}, p_j, h_{ij})). \quad (9)$$

Likewise, consider the node state x_i to be the state of the plant at the ith transmitter. If transmission is succesful, the system state evolves with stable gain $1 > \gamma_c > 0$; otherwise, it evolves with unstable gain gain $\gamma_o > 1$. We are often concerned with a quadratic cost that measures the one step distance from the origin of the plant state, which can be written as the following cost (or negative reward), i.e.

$$f_i(\mathbf{p}; \mathbf{H}, \mathbf{x}) := q_i(\mathbf{p}; \mathbf{H}, \mathbf{x})(\gamma_c x_i)^2 + (1 - q_i(\mathbf{p}; \mathbf{H}, \mathbf{x}))(\gamma_o x_i)^2. \quad (10)$$

The cost metric (10) can be used to define a utility that maximizes the expected negative cost, i. e. $u_0(\mathbf{r}) := -\mathbf{1}^T \mathbf{r}$ and constraints that impose a minimum long-term cost $\kappa_{\max}$ for each plant, i.e. $u_i(r_t) := r_i - \kappa_{\max} \leq 0$.

III. RANDOM EDGE GRAPH NEURAL NETWORKS

For the learning parametrization in (5) we introduce random edge (RE) graph neural networks (GNNs). To that end, recall the definition of $\mathbf{x} = [x_1; \ldots; x_m]$ as a vector whose entry x_i represents the state of the communication from node i to receiver $r(i)$ and reinterpret $\mathbf{x}$ as a signal supported on the nodes $i = 1, \ldots, m$. Further reinterpret the channel matrix $\mathbf{H} \in \mathbb{R}^{m \times m}$ as an adjacency matrix representation of a graph linking node i to node j. REGNNs rely on graph convolutional filters supported on the graph $\mathbf{H}$ to process some input signal $\mathbf{z} \in \mathbb{R}^m$. Formally, let $\boldsymbol{\alpha} := [\alpha_0; \ldots; \alpha_{K-1}]$ be a set of K filter coefficients and define the graph filter $\mathbf{A}(\mathbf{H})$ as a polynomial on the graph representation that is linearly applied to an input signal $\mathbf{z}$ to produce the output signal,

$$\mathbf{y} = \mathbf{A}(\mathbf{H})\mathbf{z} := \sum_{k=0}^{K-1} \alpha_k \mathbf{H}^k \mathbf{z}. \quad (11)$$

In the graph signal processing literature, the filter $\mathbf{A}(\mathbf{H}) = \sum_{k=0}^{K-1} \alpha_k \mathbf{H}^k$ is said to be a linear shift invariant filter and the

matrix $\mathbf{H}$ is a graph shift operator (GSO) [27]. If we particularize $\mathbf{H}$ to represent a cyclic graph, the operation in (11) reduces to the conventional convolution operation.

For an intuitive understanding of the graph filter in (11), consider that a single shift operation $\mathbf{H}\mathbf{z}$ will aggregate information at each node from its immediate neighbors scaled by their associated edge weights; likewise, a k-shift operation $\mathbf{H}^k\mathbf{z}$ will aggregate information from the k-hop neighborhood. This can be considered as a multi-hop message passing of local information between nodes. A benefit of larger graph filters in a GNN is that each set of node features $\mathbf{z}$ will be a generated from a more complete picture of the network. For instance, a graph filter of length $K = 2$ will generate a feature using the fading state information of a node and its neighbors, but will not directly consider the fading state information of the more distant nodes in the network—see Remark 2 for a discussion of this locality in regards to the proposed architecture.

To define a REGNN we compose L layers, each of which is itself the composition of a graph filter with a pointwise nonlinearity. Introduce then a layer index l in (11) so that we have K_l filter coefficients $\boldsymbol{\alpha}_l = [\alpha_{l0}; \ldots; \alpha_{l(K_l-1)}]$ defining graph filters $\mathbf{A}_l(\mathbf{H}) = \sum_{k=0}^{K_l-1} \alpha_{lk}\mathbf{H}^k$. We apply the filter $\mathbf{A}_l(\mathbf{H})$ to the output of layer $l-1$ to produce the layer l intermediate feature $\mathbf{y}_l = \mathbf{A}(\mathbf{H})\mathbf{z}_{l-1}$. This intermediate feature is then passed through a pointwise nonlinearity function $\boldsymbol{\sigma} : \mathbb{R}^m \to \mathbb{R}^m$ to produce the output of the lth layer as

$$\mathbf{z}_l = \boldsymbol{\sigma}\left[\mathbf{y}_l\right] = \boldsymbol{\sigma}\left[\mathbf{A}_l(\mathbf{H})\mathbf{z}_{l-1}\right] = \boldsymbol{\sigma}\left[\sum_{k=0}^{K_l-1} \alpha_{lk}\mathbf{H}^k\mathbf{z}_{l-1}\right] \quad (12)$$

A REGNN is defined by recursive application of (12). The input to this recursion is the node state vector, i.e. $\mathbf{z}_0 := \mathbf{x}$. The output is the lth layer signal $\mathbf{z}_L$. We emphasize that the nonlinear function $\boldsymbol{\sigma}$ in (12) is applied individually to each component. Namely, for any input vector $\mathbf{v}$ we must have $[\boldsymbol{\sigma}(\mathbf{v})]_i = \boldsymbol{\sigma}([\mathbf{v}]_i)$. Common choices for $\boldsymbol{\sigma}$ are rectified linear units (ReLu), absolute values, or sigmoids [21], [22].

To increase the expressive power of REGNNs we consider multiple features per layer. That is, instead of processing the output of layer $l-1$ with a single graph filter, we process it with a bank of F_l graph filters. This process generates multiple features per layer, each of which we process with a separate graph filter bank. Suppose then that the output of layer $l-1$ consists of F_{l-1} features $\mathbf{z}_l^f$. These features become inputs to layer l, each of which we process with a F_l filters $\mathbf{A}_l^{fg}(\mathbf{H})$ defined by coefficients $\boldsymbol{\alpha}_l^{fg} := [\alpha_{l0}^{fg}; \ldots; \alpha_{l(K_l-1)}^{fg}]$. Applying each of these filters to each of the input features produce the lth layer intermediate features

$$\mathbf{y}_l^{fg} = \mathbf{A}_l^{fg}(\mathbf{H})\mathbf{z}_{l-1}^f = \sum_{k=0}^{K_l-1} \alpha_{lk}^{fg}\mathbf{H}^k\mathbf{z}_{l-1}^f. \quad (13)$$

The lth layer filter bank therefore produces a total of $F_{l-1} \times F_l$ intermediate features $\mathbf{y}_l^{fg}$. To avoid exponential growth of the number of features all features $\mathbf{y}_l^{fg}$ for a given g are linearly aggregated and passed through the pointwise nonlinearity $\boldsymbol{\sigma}$ to produce the lth layer output

$$\mathbf{z}_l = \boldsymbol{\sigma}_l\left[\sum_{f=1}^{F_l} \mathbf{y}_l^{fg}\right] = \boldsymbol{\sigma}_l\left[\sum_{f=1}^{F_l} \mathbf{A}_l^{fg}(\mathbf{H})\mathbf{z}_{l-1}^f\right]. \quad (14)$$

The REGNNs we consider in this paper are defined by recursive application of (14). The input to layer $l = 1$ is the (single feature) signal $\mathbf{x} = \mathbf{z}_0^1$. The output of the REGNN is the (also single feature) signal $\mathbf{z}_L = \mathbf{z}_L^1$. For future reference we group all filter coefficients in the filter tensor $\mathbf{A} = \{\alpha_{lk}^{fg}\}_{l,f,g,k}$ and define the REGNN operator as,

$$\boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A}) = \mathbf{z}_L. \quad (15)$$

The operator in (15) is a graph neural network [21], [22] if we fix the graph $\mathbf{H}$. Here we call it a random edge (RE)GNN to emphasize that $\mathbf{H}$ is an input to the operator $\boldsymbol{\Phi}$.

Our goal is to solve (3) using as inputs the class of functions that can be represented by the REGNNs in (15). This translates to solving the optimization problem

$$\mathbf{A}^* = \operatorname{argmax} \quad u_0(\mathbf{r}),$$
$$\text{s.t.} \quad \mathbf{r} = \mathbb{E}\left[\mathbf{f}\left(\boldsymbol{\Phi}\left(\mathbf{H}, \mathbf{x}; \mathbf{A}\right); \mathbf{H}, \mathbf{x}\right)\right],$$
$$\mathbf{u}(\mathbf{r}) \geq \mathbf{0}, \quad \boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A}) \in \mathcal{P}(\mathbf{H}, \mathbf{x}). \quad (16)$$

The REGNN receives as input a random graph signal input $\mathbf{x} \sim m(\mathbf{x})$ and a random underlying graph shift operator $\mathbf{H} \sim m(\mathbf{H})$. According to (16), the filter coefficients in the tensor $\mathbf{A}$ are trained relative to the statistics of both of these quantities; the graph signal $\mathbf{x}$ and the underlying graph $\mathbf{H}$.

Notice that to solve (16) we need to specify the class of admissible GNN representations $\boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A})$. This requires specifying the number of layers, L, the number of features at each layer, F_l, and the length of the filters used at each layer, K_l. We say that L, F_l, and K_l specify the REGNN architecture. The total number of parameters that specify this architecture is $q = \sum_{l=1}^L \sum_{l=1}^L K_l \times F_l \times F_{l+1}$, which simplifies to LKF^2 if all layers use $F_l = F$ features and filters of length $K_l = K$. This number is (much) smaller than the number of parameters that would be required to train a fully connected neural network. We further point out that the feasibility condition $\boldsymbol{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A}) \in \mathcal{P}(\mathbf{H}, \mathbf{x})$ can often be easily addressed by utilizing the final output layer activation $\boldsymbol{\sigma}_L$ to project onto $\mathcal{P}(\mathbf{H}, \mathbf{x})$.

Perhaps most importantly, and as will be seen in the following section, the use of graph filters creates a permutation equivariance that matches the permutation equivariance of the optimal solution of (3). This equivariance suggests that REGNNs likely generalize across different network realizations; something we will verify in the numerical experiments in Section VI.

Remark 1 (Graph shift operator): The graph $\mathbf{H}$ is an asymmetric graph with self loops in which the weight in the edge (i, j) is the fading channel realization h_{ij}. Notice that this edge weight is *not* the strength of the channel linking node i to node j but the strength of the channel linking transmitter i to the receiver $r(j)$ associated to node j. Further observe that this is a random graph whose realizations are drawn from the distribution $m(\mathbf{H})$. The mapping of interference patterns to the graph structure can be visualized in Fig. 2.
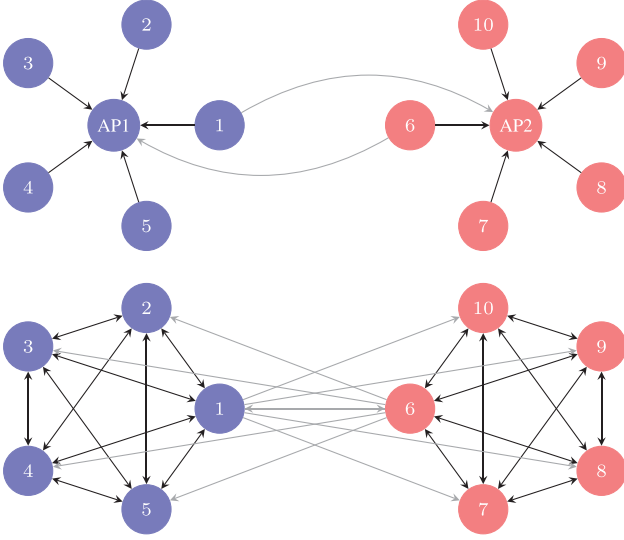
Fig. 2. Communication network and interference graph. Random edge graph neural networks (Section III) run on the interference graph (bottom), not the communication graph (top). Nodes 1-5 communicate with AP1 and nodes 6-10 with AP2 while node 1 also interferes on AP2 and node 6 on AP1. In the interference graph nodes that communicate or interfere with an AP form a clique. The APs are not nodes of the interference graph.

Remark 2 (Locality of Graph Filters): Observe in (11) that the kth filter tap scales the input by the kth order of the graph shift $\mathbf{H}$. This term reflects a k-hop shift of the elements in $\mathbf{z}$, with each hop weighted by the associated edge. As the order k increases, node states from larger neighborhoods are incorporated. Thus, the *locality* of a node, or the weight of its k hop neighborhood, provides a guide for selecting filter size K, as incorporating higher order information will have diminishing impact as the kth order neighborhood shrinks in size or weight.

Remark 3 (Convolutional Neural Networks): Just as the standard convolution operation is a particular case of the graph filter in (11) for the cyclic graph, the REGNN generalizes the standard convolutional neural network (CNN) to include random and arbitrary graph structures. CNNs have been empirically observed to be strikingly effective in many learning tasks ranging from image classification [28] to recommender systems [29]. Their success is not attributed only to their low dimensionality, but by the fact that they contain a translation equivariance property necessary for, e.g., image classification. As the REGNN is a generalization of the CNN, it is reasonable to expect that they contain similar equivariances—namely, an invariance to permutations. In Section IV-A we establish a permutation equivariance of optimal wireless resource allocation, and proceed to establish the same property in REGNNs.

Remark 4 (GNNs in Wireless Communications): The use of GNNs for learning resource allocations in wireless communications is first proposed in [23] and subsequently appears in [24], [25]. This paper extends [23] with a comprehensive evaluation of scalability, permutation equivariance, and transference. This paper differs from [24], [25] in that the latter two use a simpler GNN architecture in which the graph filters in (11) are of order $K = 1$. The use of high order linear graph filters improves the

expressive power of GNNs [30]. We also point out that [24] considers only short term resource allocation problems while both [24], [25] solve resource allocation problems without constraints. In this work we consider long term resource allocation with constraints as defined in (3).

IV. PERMUTATION INVARIANCE AND EQUIVARIANCE

Once trained, a GNN can be executed in any network independently of dimension or shape. Indeed, if we are given the filter coefficients to use in (13) we can implement the GNN for any graph $\mathbf{H}$. This is important for us because the graph $\mathbf{H}$ is randomly drawn along with input $\mathbf{x}$ from the distribution $m(\mathbf{H}, \mathbf{x})$. But it is also important because it allows execution on different networks, i.e., on networks that are drawn from a different distribution $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$. If we draw graphs $\hat{\mathbf{H}}$ and states $\hat{\mathbf{x}}$ from distribution $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$ these can be substituted into (14) to produce hidden layer signals $\hat{\mathbf{z}}_l$ and and outputs that according to the notation in (15) we can write as

$$\mathbf{\Phi}(\hat{\mathbf{H}}, \hat{\mathbf{x}}; \mathbf{A}) = \hat{\mathbf{z}}_L. \tag{17}$$

Although (17) is just a restatement of (15) with different notation, we write it to emphasize that in (17) the graph $\hat{\mathbf{H}}$ and the state $\hat{\mathbf{x}}$ are drawn from a different distribution whereas the filter tensors are the *same* in both equations. We say that the tensor that we learn for distribution $m(\mathbf{H}, \mathbf{x})$ is transferred to distribution $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$.

Transference of REGNNs can be attempted for any pair of network distributions but we do not expect good performance for transference between arbitrary network pairs. To characterize cases where we do expect good transference performance we will show here that optimal filter tensors are invariant to permutations. We begin by defining permutation matrices of dimension m as those matrices $\mathbf{\Pi}$ that belong to the set

$$\varphi = \left\{ \mathbf{\Pi} \in \{0, 1\}^{m \times m} : \mathbf{\Pi} \mathbf{1} = \mathbf{1}, \ \mathbf{\Pi}^T \mathbf{1} = \mathbf{1} \right\}. \tag{18}$$

A permutation matrix $\mathbf{\Pi}$ satisfying the conditions in (18) is one for which the product $\mathbf{\Pi}^T \mathbf{v}$ reorders the entries of any given vector $\mathbf{v}$ and in which the product $\mathbf{\Pi}^T \mathbf{M} \mathbf{\Pi}$ reorders the rows and columns of any given matrix $\mathbf{M}$. We further introduce two assumptions on the permutation invariance of the functions and constraints that define the optimal resource allocation problem in (3).

Assumption 1: The utility u_0 is permutation invariant so that for all permutation matrices $\mathbf{\Pi} \in \varphi$ it holds $u_0(\mathbf{\Pi}^T \mathbf{r}) = u_0(\mathbf{r})$.

Assumption 2: The constraint $\mathbf{u}(\mathbf{r}) \geq \mathbf{0}$ is permutation invariant in the sense that for all $\mathbf{\Pi} \in \varphi$ it holds

$$\mathbf{u}(\mathbf{r}) \geq \mathbf{0} \iff \mathbf{u}(\mathbf{\Pi}^T \mathbf{r}) \geq \mathbf{0}. \tag{19}$$

Assumption 3: The reward function $\mathbf{f}$ [cf. (1)] is permutation equivariant. I.e., for all permutation matrices $\mathbf{\Pi} \in \mathcal{P}$ it holds

$$\mathbf{f}\left(\hat{\mathbf{p}}; \hat{\mathbf{H}}, \hat{\mathbf{x}}\right) = \mathbf{\Pi}^T \mathbf{f}(\mathbf{p}; \mathbf{H}, \mathbf{x}). \tag{20}$$

where $\hat{\mathbf{H}} = \mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}$ and $\hat{\mathbf{x}} = \mathbf{\Pi}^T \mathbf{x}$ are state permutations and and $\hat{\mathbf{p}} = \mathbf{\Pi}^T \mathbf{p}$ is a resource allocation permutation.

Assumption 1 states that the utility u_0 does not change if nodes are reordered. Assumption 2 states the same for the constraint

$\mathbf{u}(\mathbf{x}) \geq \mathbf{0}$. It may be that the components of the vector function $\mathbf{u}(\mathbf{\Pi}^T \mathbf{x})$ are reordered upon permutation, but any constraint that appears in $\mathbf{u}(\mathbf{x})$ appears in $\mathbf{u}(\mathbf{\Pi}^T \mathbf{x})$. Assumption 3 requires that a reordering of the nodes results in a consistent reordering of the reward function. Since the utilities u_0 and $\mathbf{u}$ are design choices we can enforce Assumptions 1 and 2 to hold. Most usual choices for these utilities satisfy these assumptions. Assumption 3 depends on the physical model of the system. It is not a stringent requirement. All examples in Section II-A satisfy Assumption 3.

We can now state the following theorem.

Theorem 1: Consider wireless networks defined by probability distributions $m(\mathbf{H}, \mathbf{x})$ and $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$ such that there exists a permutation matrix $\mathbf{\Pi}$ such that if we define $\hat{\mathbf{H}} = \mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}$ and $\hat{\mathbf{x}} = \mathbf{\Pi}^T \mathbf{x}$ it holds

$$\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}}) = \hat{m}(\mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}, \mathbf{\Pi}^T \mathbf{x}) = m(\mathbf{H}, \mathbf{x}). \qquad (21)$$

Further assume that Assumptions 1, 2 and 3 hold. The solutions $\mathbf{A}^*$ and $\hat{\mathbf{A}}^*$ of (16) for distributions $m(\mathbf{H}, \mathbf{x})$ and $m(\hat{\mathbf{H}}, \hat{\mathbf{x}})$ are equivalent,

$$\hat{\mathbf{A}}^* \equiv \mathbf{A}^* \qquad (22)$$

Theorem 1 states permutation *invariance* of optimal filter tensors. If two networks are permutations of each other, the respective optimal REGNN filters are the same. Therefore, a REGNN that is trained over the network distribution $m(\mathbf{H}, \mathbf{x})$ can be transferred to the network distribution of a permuted network $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$ without loss of optimality. This is a useful property as it is not unreasonable to expect different large scale networks to be close to mutual permutations as we have already mentioned and illustrated in Fig. 1. This implication of Theorem 1 is explored numerically in Section VI-B.

Theorem 1 is a direct consequence of the fact that, both, REGNNs and Problem (3) are *equivariant* to permutations as we show in Sections IV-A and IV-B.

A. Permutation Equivariance of Optimal Resource Allocation

A function or policy that demonstrates permutation equivariance is one such that a permutation of inputs results in an equally permuted output. If Assumptions 1, 2 and 3 hold the following proposition asserts permutation equivariance of resource allocation policies in (3).

Proposition 1: Consider wireless networks defined by probability distributions $m(\mathbf{H}, \mathbf{x})$ and $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$ along with resource allocations $\mathbf{p}$ and $\hat{\mathbf{p}}$. Assume there exists a permutation matrix $\mathbf{\Pi}$ for which (21) holds and that for the same permutation matrix

$$\hat{\mathbf{p}}(\hat{\mathbf{H}}, \hat{\mathbf{x}}) = \mathbf{\Pi}^T \mathbf{p}(\mathbf{H}, \mathbf{x}). \qquad (23)$$

Define the respective long term rewards $\mathbf{r} = \mathbb{E}_m[\mathbf{f}(\mathbf{p}(\mathbf{H}, \mathbf{x}); \mathbf{H}, \mathbf{x})]$ and $\hat{\mathbf{r}} = \mathbb{E}_{\hat{m}}[\mathbf{f}(\hat{\mathbf{p}}(\hat{\mathbf{H}}, \hat{\mathbf{x}}); \hat{\mathbf{H}}, \hat{\mathbf{x}})]$ as per (2). If Assumptions 1–3 hold,

$$u_0(\hat{\mathbf{r}}) = u_0(\mathbf{r}), \quad \text{and} \quad \mathbf{u}(\hat{\mathbf{r}}) \geq \mathbf{0} \iff \mathbf{u}(\mathbf{r}) \geq \mathbf{0}. \qquad (24)$$

In particular, the optimal resource allocations in (3) is permutation equivariant in that for any permutation matrix $\mathbf{\Pi} \in \varphi$,

$$\mathbf{p}^*(\mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}, \mathbf{\Pi}^T \mathbf{x}) = \mathbf{\Pi}^T \mathbf{p}^*(\mathbf{H}, \mathbf{x}). \qquad (25)$$

Proof: To prove the result in (24) we prove that $\hat{\mathbf{r}} = \mathbf{\Pi}^T \mathbf{r}$. This follows readily from their definitions and the hypothesis. Begin by writing

$$\hat{\mathbf{r}} = \int \mathbf{f}\left(\hat{\mathbf{p}}\left(\hat{\mathbf{H}}, \hat{\mathbf{x}}\right); \hat{\mathbf{H}}, \hat{\mathbf{x}}\right) d\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}}), \qquad (26)$$

and observe that we have assumed $\hat{\mathbf{p}}(\hat{\mathbf{H}}, \hat{\mathbf{x}}) = \mathbf{\Pi}^T \mathbf{p}(\mathbf{H}, \mathbf{x})$. Substitute this assumption into (26) to obtain

$$\hat{\mathbf{r}} = \int \mathbf{f}\left(\mathbf{\Pi}^T \mathbf{p}(\mathbf{H}, \mathbf{x}); \hat{\mathbf{H}}, \hat{\mathbf{x}}\right) d\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}}), \qquad (27)$$

Implement the change of variables $\hat{\mathbf{H}} \to \mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}$ and $\hat{\mathbf{x}} \to \mathbf{\Pi}^T \mathbf{x}$. Since the permutation matrices are isometric, this change of variables transforms (27) into

$$\hat{\mathbf{r}} = \int \mathbf{f}\left(\mathbf{\Pi}^T \mathbf{p}(\mathbf{H}, \mathbf{x}); \mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}, \mathbf{\Pi}^T \mathbf{x}\right) d\hat{m}\left(\mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}, \mathbf{\Pi}^T \mathbf{x}\right), \qquad (28)$$

As per assumption (3) we know that the function $\mathbf{f}$ is permutation equivariant and that, therefore, $\mathbf{f}(\mathbf{\Pi}^T \mathbf{p}(\mathbf{H}, \mathbf{x}); \mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}, \mathbf{\Pi}^T \mathbf{x}) = \mathbf{\Pi}^T \mathbf{f}(\mathbf{p}(\mathbf{H}, \mathbf{x}); \mathbf{H}, \mathbf{x})$. As per (21) we know that the state distributions satify $\hat{m}(\mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}, \mathbf{\Pi}^T \mathbf{x}) = m(\mathbf{H}, \mathbf{x})$. Substituting these two facts into (28) leads to

$$\hat{\mathbf{r}} = \int \mathbf{\Pi}^T \mathbf{f}(\mathbf{p}(\mathbf{H}, \mathbf{x}); \mathbf{H}, \mathbf{x}) dm(\mathbf{H}, \mathbf{x}), \qquad (29)$$

Extracting the permutation matrix from inside the integral and using the definition of $\mathbf{r} = \mathbb{E}[\mathbf{f}(\mathbf{p}(\mathbf{H}, \mathbf{x}); \mathbf{H}, \mathbf{x})]$ leads to

$$\hat{\mathbf{r}} = \mathbf{\Pi}^T \mathbf{r}. \qquad (30)$$

Given that (30) holds, the result in (24) follows from direct application of Assumptions 1 and 2. This statement says that permutations of a network and associated permutations of resource allocation functions result in feasible rewards that attain the same utility. Therefore, (25) holds as a particular case of (30) for the optimal resource allocation functions $\mathbf{p}^*$ and $\hat{\mathbf{p}}^*$. ∎

In Proposition 1, we establish that the optimization problem in (3) is permutation equivariant and that, in particular, optimal policies are permutation equivariant. We point out that this property for the resource allocation policy follows intuition, as the labeling of the nodes is generally arbitrary–see Remark 6—and the structure of the policy should indeed reflect that. By identifying such a structural property of the policy we wish to model, we further identify a potentially lower-dimensional class of parameterizations to perform optimization over. A generic parameterization such has the FCNN may contain instances that are permutation equivariant, but it will not hold this property by default. We proceed to establish the permutation equivariance of the class of REGNNS.

B. Equivariance of Random Edge Graph Neural Networks

In comparison to its fully connected counterpart, the REGNN may appear limited by its lack of universal approximation capabilities. However, what we lose in universality we gain in *structure*. That is, in learning the weights of a REGNN in (16) we restrict our attention to a class of parameterizations that maintain desirable structure. The convolutional structure

of the REGNN architecture allows us to establish the same permutation equivariance property demonstrated for optimal resource allocation policy in Proposition 1—a permutation of the underlying graph and input signal of an REGNN will produce an equally permuted output. This result is stated formally in the following proposition.

Proposition 2: Consider graphs $\mathbf{H}$ and $\hat{\mathbf{H}}$ along with signals $\mathbf{x}$ and $\hat{\mathbf{x}}$ such that for some permutation matrix $\mathbf{\Pi}$ we have $\hat{\mathbf{H}} = \mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}$ and $\hat{\mathbf{x}} = \mathbf{\Pi}^T \mathbf{x}$. The output of a REGNN with filter tensor $\mathbf{A}$ to the pairs $(\mathbf{H}, \mathbf{x})$ and $(\hat{\mathbf{H}}, \hat{\mathbf{x}})$ are such that

$$\mathbf{\Phi}(\hat{\mathbf{H}}, \hat{\mathbf{x}}; \mathbf{A}) = \mathbf{\Pi}^T \mathbf{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A}). \tag{31}$$

Proof: This is a restatement of [30, Proposition 2]. We sketch a proof assuming the number of filters is $F_l = 1$ at each layer l for completeness. Consider first the layer $l = 1$ and take as inputs the permuted node state $\mathbf{z}_0 = \hat{\mathbf{x}} = \mathbf{\Pi}^T \mathbf{x}$ and permuted graph $\hat{\mathbf{H}} = \mathbf{\Pi}^T \mathbf{H} \mathbf{\Pi}$. The output of the first layer for the REGNN is given by (14) as

$$\mathbf{z}'_2 = \boldsymbol{\sigma}_1 [\mathbf{A}_1(\hat{\mathbf{H}}) \hat{\mathbf{x}}]. \tag{32}$$

By using the fact that $\mathbf{\Pi}^T \mathbf{\Pi} = \mathbf{\Pi} \mathbf{\Pi}^T = \mathbf{I}$ for any permutation matrix $\mathbf{\Pi}$, it follows that $\hat{\mathbf{H}}^k = \mathbf{\Pi}^T \mathbf{H}^k \mathbf{\Pi}$. By expanding the term $\mathbf{A}_1(\hat{\mathbf{H}})$ as in (12) we obtain

$$\mathbf{z}'_2 = \boldsymbol{\sigma}_1 \left[\sum_{k=0}^{K_1-1} \alpha_{1,k} \mathbf{\Pi}^T \mathbf{H}^k \mathbf{\Pi} \mathbf{\Pi}^T \mathbf{x} \right]. \tag{33}$$

$$= \boldsymbol{\sigma}_1 \left[\mathbf{\Pi}^T \sum_{k=0}^{K_1-1} \alpha_{1,k} \mathbf{H}^k \mathbf{x} \right] = \boldsymbol{\sigma}_1 [\mathbf{A}_1(\mathbf{H}) \mathbf{x}]. \tag{34}$$

Because the non-linearity $\boldsymbol{\sigma}_1(\cdot)$ is pointwise, it follows that $\mathbf{z}'_2 = \mathbf{\Pi}^T \boldsymbol{\sigma}_1(\boldsymbol{\alpha}_1 *_{\mathbf{H}} \mathbf{x}) = \mathbf{\Pi}^T \mathbf{z}_2$, where $\mathbf{z}_2$ is the output of the first layer under unpermuted inputs. As the output of a single layer is permutation equivariant to its input, it follows that the output of the composition of layers $l = 1, \ldots, L$ is permutation equivarient, i.e. $\mathbf{\Phi}(\hat{\mathbf{H}}, \hat{\mathbf{x}}; \mathbf{A}) = \mathbf{\Pi}^T \mathbf{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A})$ as stated in (31). ∎

Proposition 2 establishes the permutation equivariance of REGNNs. This structural property comes from the multiplicative relationship between the channel states $\mathbf{H}$ and the node states $\mathbf{x}$ used in the graph filter in (11). In the context of wireless networks, this implies that a relabelling or reordering of the transmitters in the network will produce an appropriately permutation of the power allocation *without any permutation of the filter weights*. This essential structural property is not satisfied by general FCNNs, in which a restructuring of the network would require an equivalent permutation of the inter-layer weights. While their full generality implies that such a permutation equivariant property *can* be satisfied by a FCNN, this property would have to be learned during training. We further note while that alternative parameterizations may also be shown to exhibit a similar permutation equivariance, our numerical study in Section VI of REGNNs shows their particular effectiveness in learning for large scale networks.

The results in Propositions 1 and 2 lead directly to our primary result in Theorem 1 as we formally show next.

Proof of Theorem 1: Suppose we are given the tensor $\mathbf{A}^*$ that is optimal for (16) for distribution $m(\mathbf{H}, \mathbf{x})$. This tensor produces a resource allocation $\mathbf{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A}^*)$ and an optimal reward $\mathbf{r}^*$. Permute the GNN inputs with $\mathbf{\Pi}$ to produce the resource allocation $\mathbf{\Phi}(\hat{\mathbf{H}}, \hat{\mathbf{x}}; \mathbf{A}^*)$, which according to the result (31) of Proposition 2 satisfies

$$\mathbf{\Phi}(\hat{\mathbf{H}}, \hat{\mathbf{x}}; \mathbf{A}^*) = \mathbf{\Pi}^T \mathbf{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A}^*). \tag{35}$$

From this fact and the hypotheses of Theorem 1, which we are trying to prove, we have that the hypotheses of Proposition 1 hold. Thus, the reward

$$\hat{\mathbf{r}} = \mathbb{E}_{\hat{m}} \left[\mathbf{f} \left(\mathbf{\Phi}(\hat{\mathbf{H}}, \hat{\mathbf{x}}; \mathbf{A}^*); \hat{\mathbf{H}}, \hat{\mathbf{x}} \right) \right] \tag{36}$$

is feasible in problem (16) with state distribution $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$ and attains utility $u(\hat{\mathbf{r}}) = u(\mathbf{r}^*)$.

Consider now the optimal tensor $\hat{\mathbf{A}}^*$ that is optimal for (16) for distribution $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$. This tensor produces a resource allocation $\mathbf{\Phi}(\hat{\mathbf{H}}, \hat{\mathbf{x}}; \hat{\mathbf{A}}^*)$ and an optimal reward $\hat{\mathbf{r}}^*$. Repeating the argument that leads to (35) and (36) we conclude that the reward

$$\mathbf{r} = \mathbb{E}_{\hat{m}} \left[\mathbf{f} \left(\mathbf{\Phi}(\mathbf{H}, \mathbf{x}; \hat{\mathbf{A}}^*); \mathbf{H}, \mathbf{x} \right) \right] \tag{37}$$

is feasible in problem (16) with state distribution $m(\mathbf{H}, \mathbf{x})$ and attains utility $u(\mathbf{r}) = u(\hat{\mathbf{r}}^*)$.

Since we know that $\hat{\mathbf{r}}^*$ is optimal for distribution $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$ and that $\hat{\mathbf{r}}$ is simply feasible we must have

$$u(\hat{\mathbf{r}}^*) \geq u(\hat{\mathbf{r}}) = u(\mathbf{r}^*), \tag{38}$$

where the second equality follows from Proposition 1 as already shown. Likewise, since $\mathbf{r}^*$ is optimal for distribution $m(\mathbf{H}, \mathbf{x})$ and $\mathbf{r}$ is feasible, it must be that

$$u(\mathbf{r}^*) \geq u(\mathbf{r}) = u(\hat{\mathbf{r}}^*). \tag{39}$$

For (38) and (39) to hold the inequalities must be equalities. Thus, $\mathbf{A}^*$ must be optimal for distribution $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$ and, conversely, $\hat{\mathbf{A}}^*$ must be optimal for distribution $m(\mathbf{H}, \mathbf{x})$. The result in (22) is implied. ∎

Remark 5 (Permutation Matrix is Unknown): The permutation equivariance results in Propositions 1 and 2, which imply the permutation invariance result of Theorem 1, do not require knowledge of the permutation $\mathbf{\Pi}$ that equalizes the distributions $m(\mathbf{H}, \mathbf{x})$ and $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$ in (21). This is worth remarking because if the permutation is known, designing operators that are permutation invariant is elementary – just undo the permutation and apply the corresponding operator. REGNNs achieve permutation invariance without knowing the permutation that relates $m(\mathbf{H}, \mathbf{x})$ and $\hat{m}(\hat{\mathbf{H}}, \hat{\mathbf{x}})$. This is as it should be to enable transference—see Section VI-B.

Remark 6 (Homogeneity vs Heterogeneity): However advantageous, permutation invariance hinders the ability to handle heterogeneous agents. Consider, for example, a weighted utility $u_0(\mathbf{r}) = \mathbf{w}^T \mathbf{r}$ that assigns different weights to the rewards of different agents but assume that otherwise, the hypotheses of Propositions 1 and 2 and Theorem 1 hold. This utility violates Assumption 1 but this assumption is not needed in Proposition 2. Thus, we know that outputs of the REGNN are permutation

equivariant [cf. (31)] and that, in particular, the resource allocations satisfy $\hat{\mathbf{r}} = \mathbf{\Pi}^T \mathbf{r}$ [cf. (30)]. We then have that $u_0(\hat{\mathbf{r}}) = u_0(\mathbf{\Pi}^T \mathbf{r}) = \mathbf{w}^T \mathbf{\Pi}^T \mathbf{r}$ which equivocates the weight assignment.

V. PRIMAL-DUAL LEARNING

To find the optimal filter tensor $\mathbf{A}^*$ we find a saddle point of the Lagrangian associated with the optimization problem in (16). To define the Lagrangian let $\boldsymbol{\lambda}$ be a multiplier associated with the constraint $\mathbf{r} = \mathbb{E}[\mathbf{f}(\mathbf{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A}); \mathbf{H}, \mathbf{x})]$ and $\boldsymbol{\mu} \geq \mathbf{0}$ a multiplier associated with the constraint $\mathbf{u}(\mathbf{r}) \geq \mathbf{0}$. Recall that the constraint $\mathbf{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A}) \in \mathcal{P}(\mathbf{H}, \mathbf{x})$ in (16) is addressed in the output layer of the REGNN and is thus not considered in the Lagrangian function. The Lagrangian of (16) is the following weighted combination of objective and constraints

$$\mathcal{L}(\mathbf{A}, \mathbf{r}, \boldsymbol{\lambda}, \boldsymbol{\mu})$$
$$= u_0(\mathbf{r}) + \boldsymbol{\lambda}^T \big[\mathbb{E}\left[\mathbf{f}\left(\mathbf{\Phi}\left(\mathbf{H}, \mathbf{x}; \mathbf{A}\right); \mathbf{H}, \mathbf{x}\right)\right] - \mathbf{r} \big] + \boldsymbol{\mu}^T \mathbf{u}(\mathbf{r}). \tag{40}$$

A saddle point of the Lagrangian in (40) is a primal dual pair $(\mathbf{A}, \mathbf{r})^\dagger, (\boldsymbol{\lambda}, \boldsymbol{\mu})^\dagger$ such that for all variables $(\mathbf{A}, \mathbf{r})$, and $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ in a sufficiently small neighborhood we have

$$\mathcal{L}\left(\mathbf{A}, \mathbf{r}, (\boldsymbol{\lambda}, \boldsymbol{\mu})^\dagger\right) \leq \mathcal{L}\left((\mathbf{A}, \mathbf{r})^\dagger, (\boldsymbol{\lambda}, \boldsymbol{\mu})^\dagger\right) \leq \mathcal{L}\left((\mathbf{A}, \mathbf{r})^\dagger, \boldsymbol{\lambda}, \boldsymbol{\mu}\right). \tag{41}$$

Namely, the saddle point is locally maximal on the primal variables and locally minimal in the dual variables. There several saddle points satisfying (41) because the Lagrangian $\mathcal{L}(\mathbf{A}, \mathbf{r}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ in (40) is associated to a nonconvex optimization problem [31, Ch. 5]. We know that one of them contains the optimal tensor $\mathbf{A}^*$. Since a global search is intractable we are going to settle for a local search. We remark that this search need not find the optimal tensor $\mathbf{A}^*$ but we have observed good empirical results.

The primal-dual method we use here alternates between gradient descent steps in the dual variables $(\boldsymbol{\lambda}, \mathbf{u})$ with gradient ascent steps in the primal variables $(\mathbf{A}, \mathbf{x})$. In specific, let k be an iteration index and ϵ denote a stepsize. The primal update on the reward function $\mathbf{r}$ is $\mathbf{r}_{k+1} = \mathbf{r}_k + \epsilon \nabla_{\mathbf{r}} \mathcal{L}(\mathbf{A}_k, \mathbf{r}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)$, which, using the explicit Lagrangian expression in (40) results in the update

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \epsilon \left[\nabla_{\mathbf{r}} u_0(\mathbf{r}_k) + [\nabla_{\mathbf{r}} \mathbf{u}(\mathbf{r}_k)] \boldsymbol{\mu}_k - \boldsymbol{\lambda}_k \right]. \tag{42}$$

Similarly, the update in the dual variable $\boldsymbol{\mu}$ takes the form $\boldsymbol{\mu}_{k+1} = [\boldsymbol{\mu}_k - \epsilon \nabla_{\boldsymbol{\mu}} \mathcal{L}(\mathbf{A}_k, \mathbf{r}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)]^+$ where we add a projection on the nonnegative orthant to account for the fact that $\boldsymbol{\mu}$ is nonnegative. Since the Lagrangian is linear in the dual variables the gradient is simple to compute and the update reduces to

$$\boldsymbol{\mu}_{k+1} = \left[\boldsymbol{\mu}_k + \epsilon \, \mathbf{u}(\mathbf{r}_k) \right]^+. \tag{43}$$

The updates in (42) and (43) are both relatively easy to carry out as they depend on the utility funtions u_0 and $\mathbf{u}$. Taking gradients with respect to the filter tensor $\mathbf{A}$ and the multiplier $\mathbf{\Lambda}$ is more challenging because of the expectation operators. The gradient

descent update with respect to $\boldsymbol{\lambda}$, for example, takes the form

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - \epsilon \, \mathbb{E}\left[\mathbf{f}\left(\mathbf{\Phi}\left(\mathbf{H}, \mathbf{x}; \mathbf{A}_k\right); \mathbf{H}, \mathbf{x}\right) - \mathbf{r}_k\right]. \tag{44}$$

This is an update that can't be computed if the distribution $m(\mathbf{H}, \mathbf{x})$ is unknown. This is a problem that is resolved by using stochastic updates in which we sample a realization $(\mathbf{H}_k, \mathbf{x}_k)$ and update $\boldsymbol{\lambda}_k$ according to

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - \epsilon \left[\mathbf{f}\left(\mathbf{\Phi}\left(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k\right); \mathbf{H}_k, \mathbf{x}_k\right) - \mathbf{r}_k\right]. \tag{45}$$

The same idea of stochastic updates is used when taking gradients with respect to the filter tensor. This yields the filter update

$$\mathbf{A}_{k+1} = \mathbf{A}_k + \epsilon \left[\nabla_{\mathbf{A}} \mathbf{f}\left(\mathbf{\Phi}\left(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k\right); \mathbf{H}_k, \mathbf{x}_k\right)\right] \boldsymbol{\lambda}_k \tag{46}$$

It is germane to point out that to implement (45) we do not need to know the model $\mathbf{f}$ mapping resource allocations and network states to user rewards. It suffices to observe the state of the network $(\mathbf{H}_k, \mathbf{x}_k)$ and implement the resource allocation $\mathbf{\Phi}(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k)$ according to the current filter tensor iterate $\mathbf{A}_k$. We can then observe the reward outcome $\mathbf{f}(\mathbf{\Phi}(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k); \mathbf{H}_k, \mathbf{x}_k)$ and use it to implement the Lagrange multiplier update in (45).

The same is not quite true for the update in (46) because it requires gradients of $\mathbf{f}$ which cannot be directly queried by probing the system. This is a challenge that also arises in policy gradient methods where it is resolved with the introduction of randomized policies [32]. Mimicking that approach we reinterpret $\mathbf{\Phi}(\mathbf{H}, \mathbf{x}; \mathbf{A})$ as the parameter of probability distribution $\mathbf{\Psi}(\mathbf{H}, \mathbf{x}; \mathbf{A})$. A likelihood ratio identity allows us to express the gradient in (46) as

$$\nabla_{\mathbf{A}} \mathbb{E}\left[\mathbf{f}\left(\mathbf{\Phi}\left(\mathbf{H}, \mathbf{x}; \mathbf{A}_k\right); \mathbf{H}, \mathbf{x}\right)\right]$$
$$= \mathbb{E}\left[\mathbf{f}\left(\mathbf{\Phi}\left(\mathbf{H}, \mathbf{x}; \mathbf{A}_k\right); \mathbf{H}, \mathbf{x}\right) \nabla_{\mathbf{A}} \log \mathbf{\Psi}\left(\mathbf{H}, \mathbf{x}; \mathbf{A}_k\right)^T\right]. \tag{47}$$

The substitution in (47) is useful in that it replaces the gradient of the expectation of $\mathbf{f}$ with the expectation of the gradient of $\log \mathbf{\Psi}$. As in the stochastic update in (45), we can estimate the gradient with sampled realization $(\mathbf{H}_k, \mathbf{x}_k)$, while the gradient of $\log \mathbf{\Psi}$ can be calculated assuming a given distribution $\mathbf{\Psi}(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k)$. This results in the stochastic policy for the filter tensor $\mathbf{A}$

$$\mathbf{A}_{k+1} = \mathbf{A}_k +$$
$$\epsilon \left[\mathbf{f}\left(\mathbf{\Phi}\left(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k\right); \mathbf{H}_k, \mathbf{x}_k\right) \nabla_{\mathbf{A}} \log \mathbf{\Psi}\left(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k\right)^T \boldsymbol{\lambda}_k\right]. \tag{48}$$

We stress that (48) is model-free because, like (45), it is computed only by probing $\mathbf{f}(\mathbf{\Phi}(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k); \mathbf{H}_k, \mathbf{x}_k)$ and thus does not need explicit knowledge of $\mathbf{f}$ or the state distribution $m(\mathbf{H}, \mathbf{x})$.

We train REGNNs using the updates in (42), (43), (45), and (48). The resulting scheme is summarized in Algorithm 1. To initialize the training process we specify the REGNN architecture. We use the shorthand REGNN$[L, (F_1, K_l), \dots, (F_L, K_L)]$ to signify a REGNN with L layers in which the lth layer contains F_l features generated by filters of length K_l. After initializing REGNN filter tensor $\mathbf{A}_0$ and other primal and dual variables in Step 2, we perform our learning iterations in Step 3.

Algorithm 1: Primal-Dual REGNN Training.

1:    **Parameters:** REGNN$[L, (F_1, K_l), \ldots, (F_L, K_L)]$
2:    **Input:** Initial states $\mathbf{A}_0, \mathbf{r}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0$
3:    **for** $k = 0, 1, 2, \ldots$ **do** {main loop}
4:      Sample states $(\mathbf{H}_k, \mathbf{x}_k) \sim m(\mathbf{H}, \mathbf{x})$.
      Probe $\mathbf{f}(\boldsymbol{\Phi}(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k); \mathbf{H}_k, \mathbf{x}_k)$.
5:      Update primal and dual variables [(42), (43), (45), (48)]

$$\mathbf{r}_k + \epsilon \left[\nabla_{\mathbf{r}} u_0(\mathbf{r}_k) + [\nabla_{\mathbf{r}} \mathbf{u}(\mathbf{r}_k)] \boldsymbol{\mu}_k - \boldsymbol{\lambda}_k \right]$$

$$[\boldsymbol{\mu}_k + \epsilon \mathbf{u}(\mathbf{r}_k)]^+$$

$$\boldsymbol{\lambda}_k - \epsilon \left[\mathbf{f} \left(\boldsymbol{\Phi} \left(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k \right); \mathbf{H}_k, \mathbf{x}_k \right) - \mathbf{r}_k \right],$$

$$\mathbf{A}_k + \epsilon \big[\mathbf{f} \left(\boldsymbol{\Phi} \left(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k \right); \mathbf{H}_k, \mathbf{x}_k \right)$$

$$\nabla_{\mathbf{A}} \log \boldsymbol{\Psi}(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k)^T \boldsymbol{\lambda}_k \big]$$

6:    **end for**

Each learning iteration k consists of first computing of drawing samples of states $\mathbf{H}$ and $\mathbf{x}$ and probing the system $\mathbf{f}(\boldsymbol{\Phi}(\mathbf{H}_k, \mathbf{x}_k; \mathbf{A}_k); \mathbf{H}_k, \mathbf{x}_k)$ using the current REGNN policy specified by $\mathbf{A}_k$ in Step 4. In Step 5 we perform the primal-dual gradient updates in (42), (43), (45), and (48) using the probed performance. The process is repeated until convergence.

VI. NUMERICAL RESULTS

In this section, we provide a numerical study of the performance of resource allocation policies that parameterized with REGNNs and trained with the model-free primal-dual learning method. We simulate the performance of the policy on a number of canonical resource allocation functions that take the form of (3) and compare against existing heuristic approaches. Where applicable, we point out the compared heuristics that rely on accurate model knowledge to be implemented, which as discussed in Section V, is *not* required to implement the primal-dual learning method. We study the canonical problem of binary power control between m transmitter/receiver pairs over an AWGN channel with noise power σ^2 and interference—see the first example in Section II-A for a discussion of this problem. In addition to maximizing the sum-rate capacity, a practical constraint of interest is a maximum average power budget $P_{\max}$ to be shared between transmitters connected to a common power supply. The complete resource allocation problem can be written as

$$P^* : = \max_{\mathbf{p}(\mathbf{H}, \mathbf{x}), \mathbf{r}} \sum_{i=1}^{m} r_i,$$

$$\text{s.t. } r_i = \mathbb{E}\left[\log \left(1 + \frac{|h_{ii}|^2 p_i(\mathbf{H})}{\sigma^2 + \sum_{j \neq i} |h_{ji}|^2 p_j(\mathbf{H})} \right) \right],$$

$$\mathbb{E}\left[\mathbf{1}^T \mathbf{p}(\mathbf{H}, \mathbf{x}) \right] \leq P_{\max}, \quad \mathbf{p}(\mathbf{H}, \mathbf{x}) \in \{0, p_0\}^m. \tag{49}$$

As discussed in Section II-A, the problem in (49) does not utilize any node state $\mathbf{x}$ and associated cost constraint. This is however an instructive problem to study, as it is well studied and has

numerous developed heuristic solutions with which the compare as baselines. Observe also that the power allocation is a binary selection of transmitting with power p_0 or not transmitting.

A. Ad-hoc Networks

We begin studying the performance of the model-free training of an REGNN in the wireless ad-hoc, or paired network. For all sets of simulations, we construct the ad-hoc wireless network as follows. For a set of m pairs, we construct a random geometric graph by dropping transmitter i uniformly at random at location $\mathbf{t}_i \in [-m, m]^2$, with its paired receiver at location $\mathbf{r}_i \in [\mathbf{t}_i - m/4, \mathbf{t}_i + m/4]^2$ around its paired transmitter—see, e.g., Fig. 1(a) for an example. Given the geometric placements, the complete fading channel state between transmitter i and receiver j is composed of $h_{ij} = h_{ij}^p h_{ij}^f$, where h_{ij}^p is a constant path-loss gain and h_{ij}^f is the time varying fast fading. The path loss is related to the geometric distance as $h_{ij}^p = \|\mathbf{t}_i - \mathbf{r}_j\|^{-2.2}$ and the fast fading h_{ij}^f is drawn randomly from a standard Rayleigh distribution at each scheduling cycle and the noise power is initially fixed at $\sigma^2 = 1$.

In employing the primal dual learning method in Algorithm 1, we consider the model free version in which gradients are estimated via the policy gradient approximation. We construct an REGNN architecture with $L = 8$ hidden layers, each with $F_l = 1$ graph filters of length $K_l = 5$ and a standard ReLu non-linear activation function i.e. $\boldsymbol{\sigma}(\mathbf{z}) = [\mathbf{z}]_+$. The final layer is passed through a sigmoid function to normalize the outputs, which are then used as the parameter of a the policy distribution $\boldsymbol{\Psi}(\mathbf{H}, \mathbf{x}; \mathbf{A})$—chosen here as a Bernoulli distribution. The primal dual method is performed with a geometrically decaying step size for dual updates and the ADAM optimizer [33] for the primal updates.

In general, we make our comparisons against existing heuristic methods for solving (49). We primarily consider (i) the popular WMMSE heuristic [6] as a baseline, while also making comparisons against naive heuristics that either (ii) assign equal power $P_{\max}/m$ to all users or (iii) randomly select $P_{\max}/p_0$ users to transmit with full power. Furthermore, we simulate the learning and performance of the convolutional REGNN architecture to a fully connected neural network (FCNN) for medium scale networks. In Fig. 3, we show the performance, or sum-capacity, achieved throughout the learning process of the REGNN and FCNN trained with the primal-dual learning method and the performance of the three heuristic baselines for a medium scale wireless system with $m = 20$ pairs. It can be observed that both the REGNN and FCNN outperform the performance of WMMSE for the medium scale system. We stress that this performance is obtained by the NNs using the model-free gradients, meaning that knowledge of the capacity function was not assumed. Explicit knowledge of capacity functions is needed, however, for the WMMSE algorithm. We further point out that, although there is generally no clear way to interpret resource allocation decisions made my a learning model with which we can attribute the gains relative to WMMSE, the WMMSE method does not explicitly solve the constrained resource allocation problem in (49) as can be done by the
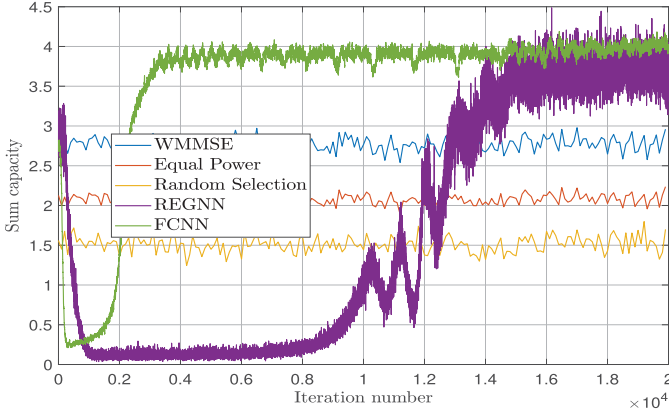
Fig. 3. Performance comparison during training of REGNN for $m = 20$ pairs. With only $q = 40$ parameters, the REGNN outperforms the WMMSE algorithm and matches the performance of the FCNN.
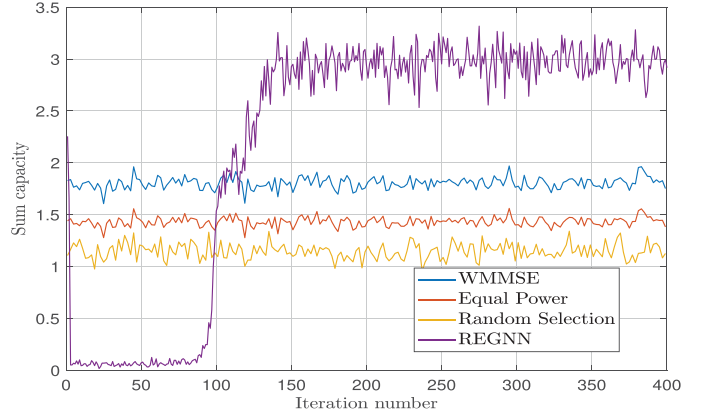

Fig. 4. Performance comparison during training of REGNN for $m = 50$ pairs. With only $q = 40$ parameters, the REGNN strongly outperforms the WMMSE algorithm.
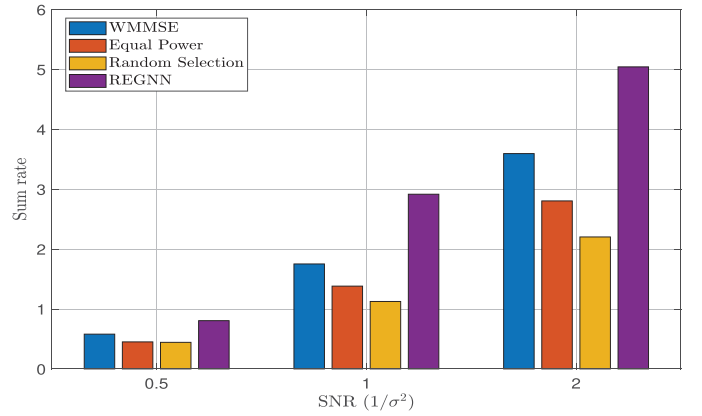

Fig. 5. Performance comparison of an REGNN against baselines in different SNR regimes. The average SNR is controlled by the AWGN noise parameter $1/\sigma^2$, larger values reflect larger average SNR. The REGNN has bigger gain over WMMSE in higher SNR regimes.

primal-dual learning method. In these simulations, satisfaction of the sum-power constraint in (49) was enforced in WMMSE by subtracting power equally across selected transmitters.

Observe that the REGNN, with only $q = 40$ parameters, matched the performance of the FCNN with two fully connected layers of size 64 and 32 for a total of $q = 20 \times 64 + 64 \times 32 + 32 \times 20 \approx 4000$ parameters—a 100 factor increase than that used by the REGNN. We also point out that, while 20000 iterations are needed for convergence here, and it can be further observed in Fig. 3 that the REGNN requires more iterations than the FCNN to achieve stronger performance. We first stress that the issue of number of iterations until convergence is not critical in this setting, as the training is done *offline*; the runtime complexity of the REGNN is lower than that of a FCNN. The proceeding simulation experiments demonstrate that the REGNN is capable of scaling to larger systems. In Section VI-B, we will demonstrate how the REGNN is moreover able transfer to networks of different configurations and sizes. This could not be possible with a FCNN, whose input dimension is of a fixed size. Therefore, while in the particular case shown in Fig. 3, it takes longer to train the REGNN, it can be considered overall much slower to train a FCNN, as it necessarily needs to be retrained for each new configuration of the network.

In Fig. 4, we show the performance while learning a REGNN in a larger scale system with $m = 50$ transmitter/receiver pairs. At this scale, the parameter dimension of the FCNN makes it challenging to train; the input dimension of channel states is 2500. Here, we see that the learned REGNN substantially outperforms all three heuristics, including WMMSE. Observe that while WMMSE achieves a sum-capacity of roughly 2.7 in the medium scale system, the algorithm performs worse when more transmitters are added, obtaining a sum-rate of only 1.8. The REGNN, meanwhile, is able to still achieve a sum-rate of 3.0, all while only learning 40 parameters. In Fig. 5, we evaluate the gains of the learned REGNN over the baselines in different SNR regimes. To vary the average SNR, we set the AWGN noise parameter $1/\sigma^2$ to vary between $\{0.5, 1, 2\}$, where a larger value reflects a higher average SNR. As can be seen, the REGNN

has substantial gain over all methods in the middle and larger SNR regimes, but only matches the performance of WMMSE when the average SNR value is small. This follows from the fact that WMMSE is known to approach the optimal value of the sum-rate maximization problem as the SNR decreases [6] and corroborates similar findings in comparisons with other machine learning-based methods, e.g. [18].

B. Transference

As previously discussed, we are interested in exploring the generalization abilities of an REGNN learned over some fixed network. Recall that the filter-bank structure of an REGNN in (15) allows the same neural network to receive inputs of varying input dimension, or network size. The permutation invariance result in Theorem 1 demonstrates that an REGNN trained on one network performs as well on permutation of that network. As we don't expect to see exact permutations in practice, here we numerically investigate the transference capabilities on randomly drawn networks of fixed density. Consider the REGNN leaned in the previous experiment in Fig. 4. As an instructive example, consider another randomly drawn network of 50 pairs
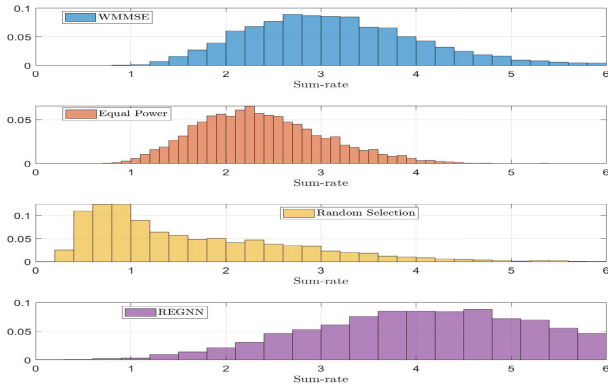
Fig. 6. Performance comparison of an REGNN that was trained in Fig. 4 in another randomly drawn network of equal size. Presented is the empirical distribution of the sum-rate achieved over many random iterations for all heuristic methods.
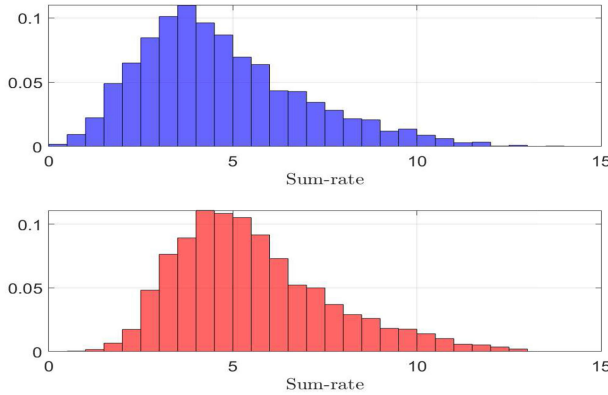

Fig. 7. Empirical histogram of sum rates obtained by (top, blue) REGNN trained on network of size $m = 50$ and (bottom, red) REGNN trained on network of size $m' = 75$ on 50 randomly drawn networks of size of $m' = 75$. The REGNN trained on the smaller network closely matches the performance of the REGNN trained on the larger network.
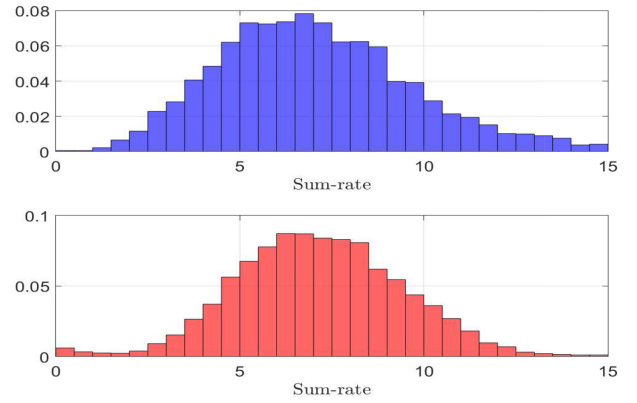

Fig. 8. Empirical histogram of sum rates obtained by (top, blue) REGNN trained on network of size $m = 50$ and (bottom, red) REGNN trained on network of size $m' = 100$ on 50 randomly drawn networks of size of $m' = 100$. The REGNN trained on the smaller network closely matches the performance of the REGNN trained on the larger network.
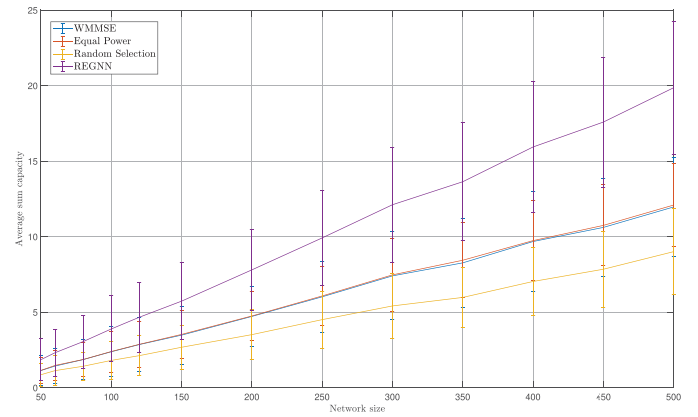

Fig. 9. Performance of REGNN trained in Fig. 4 in randomly drawn networks of varying size. From networks of size $m' = 50$ to 500, the REGNN is able to outperform the heuristic methods.

as shown in Fig. 6. The performance of the REGNN trained in Fig. 4 over many random iterations is shown here compared to the heuristics as an empirical histogram of sum-rates over all random iterations. We see that the same parameterization learned for one network performs well with another network. Intuitively speaking, this relates to the stability and permutation equivariance of GNNs because random networks of size 50 may be close to each other in expectation.

Another comparison of interest here is the relative performance of a REGNN trained on a network of size $m = 50$ with an REGNN trained on a network of $m' > m$. In Fig. 7 we show a histogram of the sum-rate performance over 50 randomly generated networks of size $m' = 75$ and $m' = 100$. For a set of m' pairs, we construct a random geometric graph by dropping transmitter i uniformly at random at location $\mathbf{t}_i \in [-m\sqrt{m'/m}, m\sqrt{m'/m}]^2$, with its paired receiver at location $\mathbf{r}_i \in [\mathbf{t}_i - m/4, \mathbf{t}_i + m/4]^2$ around its paired transmitter. This is done to keep the density of the network constant as the number of transceiver pairs grows.

The performance for each random network is itself evaluated over 100 separate fast fading samples. As can be seen, the

performance of the REGNN trained on the smaller network of size $m = 50$ almost matches the performance of an REGNN trained on a network of size 75. The same procedure is performed for networks of size $m' = 100$. In Fig. 8, we show the performance of the REGNN trained on a network of size 50 against the performance of a network trained on a network of size 100 on random networks of size 100. Again, the performance of the REGNN trained on the smaller network only slightly degrades relative to the REGNN trained on the larger network. This highlights a potential to train REGNNs on smaller networks to later be implemented on larger networks. We point out that this is a powerful property for practical learning for such systems, as we can potentially train our neural networks on smaller systems when larger networks are either unavailable during training or when computational expense is prohibitive.

To fully explore these capabilities for increasingly large networks, we again use the REGNN trained in Fig. 4 in random wireless networks of increasing size. Note that, as we increase the size of the networks, the density of the network remains constant so that the statistics of the channel conditions are the same. In Fig. 9, we show the average sum-rate achieved by the
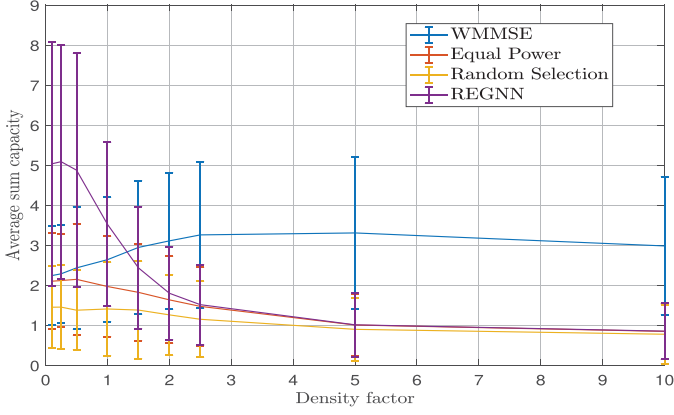
Fig. 10. Performance of REGNN trained in Fig. 4 in randomly drawn networks of varying densities from factors ranging from $r = 0.1$ to 10. As the density of the network increases, the REGNN is unable to match the performance of the WMMSE algorithm.

REGNN over many random iterations for networks of increasing size m', where the geometric configurations generated using the fixed-density random geometric graph as done previously. We observe that, even as the network size increases, the same REGNN is able to outperform the heuristic methods.

As a final numerical study for the pairwise network, we compare the performance of the REGNN trained on a fixed network of size $m = 50$ in new random networks of equivalent number of pairs but varying density. In these experiments, we draw random geometric graphs with some density factor r by dropping transmitter i uniformly at random at location $\mathbf{t}_i \in [-r^{-1}m\sqrt{m'/m}, r^{-1}m\sqrt{m'/m}]^2$, with its paired receiver at location $\mathbf{r}_i \in [\mathbf{t}_i - m/4, \mathbf{t}_i + m/4]^2$ around its paired transmitter. In this manner, as the density factor r increases, the physical space of the network gets smaller and thus more dense. In Fig. 10, we show the average sum-rate achieved by the REGNN over many random iterations for networks of increasing densities r. We observe that, for wireless networks of equal or less density than the one used for training, the REGNN has strong performance relative to the heuristics. However, as the networks more dense, the REGNN is unable to match the performance of WMMSE. This results follows from the fact that the statistics of data seen in training begins to vary more greatly from that seen in execution time as the networks increase in density. Indeed, as the transmitters become closer together, the path-loss component of the fading state decreases and the interference grows.

C. Incorporating User Demand

We proceed to perform simulations an extension to the binary control problem in the paired ad-hoc network studied in the Section VI-A—namely binary power control with varying user demand. The problem was previously discussed in the second example of Section II-A. For this setting, we assume that each transmitter additionally maintains a local state x_i that reflects its current capacity demand, such as a packet arrival rate. Such a problem is highly relevant in, e.g. sensor networks and robotics applications. The resource allocation problem is closely related to that in (49), but with an additional constraint the ergodic
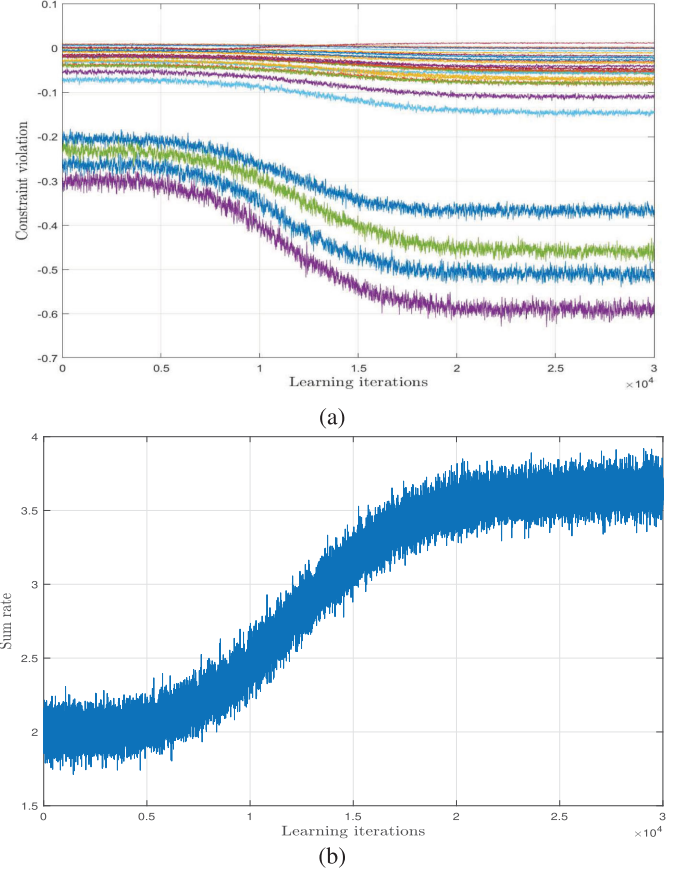

(a)

(b)

Fig. 11. Convergence of training of an REGNN for the wireless sensor network problem with $m = 30$ sensors/transmitters. In the top figure, we show the constraint violation for each of the transmitters converges to a feasible solution. In the bottom figure, we show the objective function converge to a local maximum.

capacity achieved by each transmitter must exceed the average collection rate of its associated sensor. We may write this problem as

$$P^* := \max_{\mathbf{p}(\mathbf{H},\mathbf{x}),\mathbf{r}} \sum_{i=1}^{m} r_i,$$

$$\text{s.t. } r_i = \mathbb{E}\left[\log\left(1 + \frac{|h_{ii}|^2 p_i(\mathbf{H})}{1 + \sum_{j \neq i} |h_{ji}|^2 p_j(\mathbf{H})}\right)\right],$$

$$\mathbb{E}\left[\mathbf{x}\right] \leq \mathbf{r}, \quad \mathbf{p}(\mathbf{H},\mathbf{x}) \in \{0, p_0\}^m. \tag{50}$$

The problem in (50) contains an additional complexity in the ergodic constraint, which must be independently satisfied by each transmitter. The optimal resource allocation policy is which obtains sufficient capacity is achieved by each transceiver pair in expectation, while then maximizing the sum-rate achieved over the network.

We perform the primal-dual learning method to train a REGNN in a system with $m = 30$ transmitter/receiver pairs, who are placed randomly as in previous simulations. The collection rates $\boldsymbol{\eta}$ are drawn from a exponential distribution with mean 0.05. We train a REGNN with $L = 10$ layers, each with $F_l = 1$ filters of length $K_l = 5$. In Fig. 11 we show the performance of the learning procedure. In Fig. 11(a), we show the constraint
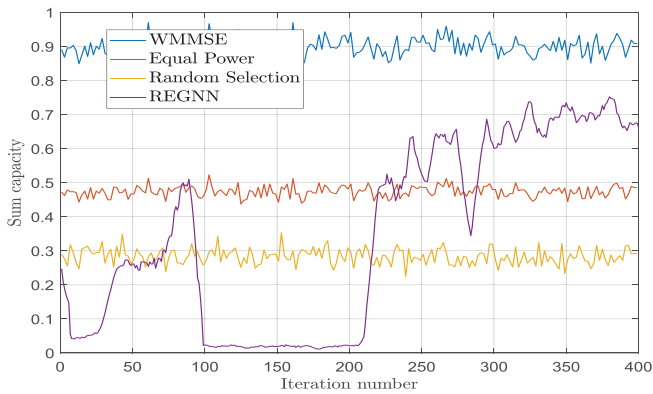
Fig. 12. Performance comparison during training of REGNN for mutli-cell interference network with $m = 50$ users and $n = 5$ base stations. With $q = 40$ parameters, the REGNN outperforms the model-free heuristics but does not quite meet the performance of WMMSE, which utilizes model information in its implementation.

violation for all 30 sensors. with a negative value of $\mathbb{E}[\mathbf{x}] - \mathbf{r}$ signifying a satisfaction of the capacity constraint, i.e. the sensor is transmitting data faster that it collects. While we may observe a wide variance in the capacities achieved by different sensors, a close examination of Fig. 11(a) shows that all but 1 sensor achieves constraint satisfaction (constraint satisfaction can be seen with a negative constraint violation value). Likewise, in Fig. 11(b), we show the REGNN-based resource allocation policy achieve an overall performance converges to local optimum as the rate constraints are being satisfied.

D. Multi-Cell Interference Network

In this section, we consider a variation of the network architecture previously considered known as a multi-cell interference network. In contrast to the pair-wise setting, in the mutli-cell network there exist n receivers—or cellular base stations—who service the transmissions of a total m cellular users, which we assume are distributed evenly amongst the base stations. An example of such a multi-cell configuration is provided in Fig. 1(b) for $n = 20$ base stations, marked with large circles, covering a variety of cellular users, marked with smaller circles. The settings here is instructive not only in the real world practicality of its setting, but in its tendency to scale largely as the number of base stations or number of users grows. In Fig. 12, we show the performance obtained by the REGNN during training compared to the heuristic methods on a network of $n = 5$ cells with a total of $m = 50$ users. Here we see that the REGNN almost matches the performance of WMMSE without requiring model knowledge in its training or execution. As before, we study the transference properties as the number of cells grow. In Fig. 13, we plot the performance of the REGNN trained in Fig. 12 on networks of increasing size by increasing the number of cells. We observe that the performance of the REGNN scales well up to $n = 30$ cells ($m = 150$ users) and matches the performance of WMMSE without any model knowledge being utilized.
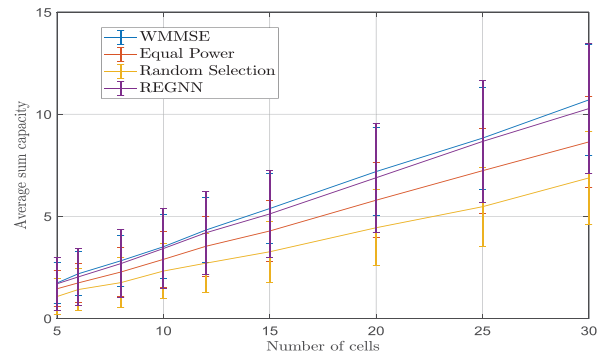

Fig. 13. Performance of REGNN trained in Fig. 12 in randomly drawn multi-cell networks of varying size. From networks of size 5 to 50 cells, the REGNN matches the performance of the best performing heuristic method.

VII. CONCLUSION

We consider the problem of learning optimal resource allocation policies in wireless networks. The resource allocation problem takes the form of constrained statistical learning, which can be addressed by training a parameterization of the resource allocation policy using a model-free, primal-dual learning method. Fully connected neural networks are unsuitable parameterizations for large scale problems due to their prohibitively large parameter dimension. Given the randomly varying graph structure of fading channel states in a wireless network, we propose the use of random edge graph neural networks (REGNNs) to parameterize the resource allocation policy. Such a neural network structure has significantly smaller parameter dimension that does not scale with the size of the wireless network. Moreover, we demonstrate that such policies are permutation equivariant, and can thus achieve similar performance on networks that are close to permutations of one another. We demonstrate in a series of numerical simulations how the REGNN is an effective parameterization for resource allocation policies for large scale wireless networks, both in learning strong performing policies and for such policies to transfer well to networks of larger size.

REFERENCES

[1] A. Ribeiro, "Optimal resource allocation in wireless communication and networking," *EURASIP J. Wireless Commun. Netw.*, vol. 2012, no. 1, 2012, Art. no. 272.

[2] W. Yu and R. Lui, "Dual methods for nonconvex spectrum optimization of multicarrier systems," *IEEE Trans. Commun.*, vol. 54, no. 7, pp. 1310–1322, Jul. 2006.

[3] J. Zhang and D. Zheng, "A stochastic primal-dual algorithm for joint flow control and mac design in multi-hop wireless networks," in *Proc. 40th Annu. Conf. Inf. Sci. Syst.*, 2006, pp. 339–344.

[4] K. Gatsis, M. Pajic, A. Ribeiro, and G. J. Pappas, "Opportunistic control over shared wireless channels," *IEEE Trans. Autom. Control*, vol. 60, no. 12, pp. 3140–3155, Dec. 2015.

[5] X. Wang, T. Chen, X. Chen, X. Zhou, and G. B. Giannakis, "Dynamic resource allocation for smart-grid powered MIMO downlink transmissions," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3354–3365, Dec. 2016.

[6] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2011, pp. 3060–3063.

[7] C. S. Chen, K. W. Shum, and C. W. Sung, "Round-robin power control for the weighted sum rate maximisation of wireless networks over multiple interfering links," *Eur. Trans. Telecommun.*, vol. 22, no. 8, pp. 458–470, 2011.

[8] X. Wu *et al.*, "FlashLinQ: A synchronous distributed scheduler for peer-to-peer ad hoc networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1215–1228, Aug. 2013.

[9] N. Naderializadeh and A. S. Avestimehr, "ITLinQ: A new approach for spectrum sharing in device-to-device communication systems," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1139–1151, Jun. 2014.

[10] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for wireless resource management," *IEEE Trans. Signal Process.*, vol. 66, no. 20, pp. 5438–5453, Oct. 2018.

[11] L. Lei, L. You, G. Dai, T. X. Vu, D. Yuan, and S. Chatzinotas, "A deep learning approach for optimizing content delivering in cache-enabled HetNet," in *Proc. Int. Symp. Wireless Commun. Syst.*, 2017, pp. 449–453.

[12] D. Xu, X. Chen, C. Wu, S. Zhang, S. Xu, and S. Cao, "Energy-efficient subchannel and power allocation for hetnets based on convolutional neural network," *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, Kuala Lumpur, Malaysia, 2019, pp. 1–5, doi: 10.1109/VTCSpring.2019.8746493.

[13] T. V. Chien, E. Björnson, and E. G. Larsson, "Sum spectral efficiency maximization in massive mimo systems: Benefits from deep learning," *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, 2019, pp. 1–6, doi: 10.1109/ICC.2019.8761234.

[14] P. de Kerret, D. Gesbert, and M. Filippone, "Team deep neural networks for interference channels," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2018, pp. 1–6.

[15] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, "A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[16] W. Lee, M. Kim, and D.-H. Cho, "Deep power control: Transmit power control scheme based on convolutional neural network," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1276–1279, Jun. 2018.

[17] M. Eisen, C. Zhang, L. F. Chamon, D. D. Lee, and A. Ribeiro, "Learning optimal resource allocations in wireless systems," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2775–2790, May 2019.

[18] F. Liang, C. Shen, W. Yu, and F. Wu, "Towards optimal power control via ensembling deep neural networks," *IEEE Trans. Wireless Commun.*, vol. 68, no. 3, pp. 1760–1776, Mar. 2020, doi: 10.1109/TCOMM.2019.2957482.

[19] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," 2019, *arXiv:1901.07159*.

[20] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1248–1261, Jun. 2019.

[21] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*.

[22] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 1034–1049, Feb. 2019.

[23] M. Eisen and A. Ribeiro, "Large scale wireless power allocation with graph neural networks," in *Proc. IEEE 20th Int. Workshop Signal Process. Advances Wireless Commun.*, 2019, pp. 1–5.

[24] M. Lee, G. Yu, and G. Y. Li, "Graph embedding based wireless link scheduling with few training samples," 2019, *arXiv:1906.02871*.

[25] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "A graph neural network approach for scalable wireless power control," *2019 IEEE Globecom Workshops (GC Wkshps)*, Waikoloa, HI, USA, 2019, pp. 1–6, doi: 10.1109/GCWkshps45667.2019.9024538.

[26] X. Wang and L. Cai, "Proportional fair scheduling in hierarchical modulation aided wireless networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 4, pp. 1584–1593, Apr. 2013.

[27] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, 2014.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[29] H.-T. Cheng *et al.*, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.

[30] F. Gama, J. Bruno, and A. Ribeiro, "Stability properties of graph neural networks," 2019, *arXiv:1905.04497*.

[31] D. P. Bertsekas, *Nonlinear Programming*. New York, NY, USA: Taylor & Francis, 1997.

[32] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Advances Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–13.

Mark Eisen (Member, IEEE) received the Ph.D in electrical engineering and Masters in statistics from the University of Pennsylvania in 2019. Since August 2019, he has been working as a Research Scientist at Intel Labs in Hillsboro, OR. His research interests include machine learning, wireless communications, networked control systems, and statistical optimization. In the summer of 2013, he was a Research Intern with the Institute for Mathematics and its Applications at the University of Minnesota and in the summer of 2018, was a Research Intern at Intel Corporation. Dr. Eisen was a recipient of the Outstanding Student Presentation at the 2014 Joint Mathematics Meeting, as well as the recipient of the 2016 Penn Outstanding Undergraduate Research Mentor Award.

Alejandro Ribeiro received the B.Sc. degree in electrical engineering from the Universidad de la Republica Oriental del Uruguay, Montevideo, in 1998 and the M.Sc. and Ph.D. degree in electrical engineering from the Department of Electrical and Computer Engineering, the University of Minnesota, Minneapolis in 2005 and 2007. From 1998 to 2003, he was a member of the technical staff at Bellsouth Montevideo. After his M.Sc. and Ph.D studies in 2008, he joined the University of Pennsylvania (Penn), Philadelphia, where he is currently the Rosenbluth Associate Professor at the Department of Electrical and Systems Engineering. His research interests are in the applications of statistical signal processing to the study of networks and networked phenomena. His focus is on structured representations of networked data structures, graph signal processing, network optimization, robot teams, and networked control. Dr. Ribeiro received the 2014 O. Hugo Schuck Best Paper Award, and paper awards at CDC 2017, 2016 SSP Workshop, 2016 SAM Workshop, 2015 Asilomar SSC Conference, ACC 2013, ICASSP 2006, and ICASSP 2005. His teaching has been recognized with the 2017 Lindback Award for Distinguished Teaching and the 2012 S. Reid Warren, Jr. Award presented by Penn's undergraduate student body for outstanding teaching. Dr. Ribeiro is a Fulbright scholar class of 2003 and a Penn Fellow class of 2015.