Resource Allocation in Wireless Communication Networks

Zhiyang Wang and Alejandro Ribeiro

October 20, 2020

In a wireless communication system, information is transmitted in the form of an electromagnetic signal between a source and a destination. The feature that determines the quality of a communication link is the signal to noise ratio (SNR). In turn, the SNR of a channel is the result of multiplying the transmit power of the source by a propagation loss. This propagation loss is what we call a wireless channel. The problem of allocating resources in wireless communications is the problem of choosing transmit powers as a function of channel strength with the goal of optimizing some performance metric that is of interest to end users.

Mathematically, the allocation of resources in wireless communications results in constrained optimization problems. Thus, in principle at least, all that is required to find an optimal allocation of resources is to find the solution of a constrained optimization program. This is, however, impossible except in a few exceptional cases. In this lab we will explore the use of graph neural networks (GNNs) to find approximate solutions to the optimal allocation of resources in wireless communication systems.

1 Point to Point Wireless Communications

We begin by building a model of a point-to-point communications channel. Consider time slots indexed by t and suppose that at time t the source selects transmit power $p(t)$. The signal propagates through a channel with attenuation coefficient $h(t)$. This is called a fading coefficient in wireless

communications. The received power $p_{\mathrm{R}}(t)$ is therefore given by the product

$$p_{\mathrm{R}}(t) = h(t)p(t). \tag{1}$$

This product determines the channel's SNR through comparison with the noise level N_0. In turn, the SNR is translated to a performance measure $c(t)$ according to a function f,

$$c(t) = f\left(\frac{h(t)p(t)}{N_0}\right). \tag{2}$$

The most distinctive feature of a wireless channel is that the fading coefficient $h(t)$ in (1) exhibits rapid random variations. It is therefore customary to consider $h(t)$ as a random variable with distribution $m(h)$. A consequence of the rapid variations in the value of $h(t)$ is that the performance measure $c(t)$ in (2) is instantaneous. It is not reflective of the experience of the system's users. The quantity that reflects user experience is the average of $c(t)$ over a period of time. This is the average reward c which is defined as the limit

$$c \;:=\; \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{\infty} c(t) \;=\; \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{\infty} f\left(\frac{h(t)p(t)}{N_0}\right). \tag{3}$$

An interesting observation is that the sum in (3) can be rewritten as an expectation over the probability distribution of the wireless channel. To do so we consider channel realizations h drawn from the channel probability distribution $m(h)$. Whenever we observe channel h we allocate power $p(h)$. Then, the performance of interest is equivalently written as

$$c \;:=\; \mathbb{E}\left[f\left(\frac{hp(h)}{N_0}\right)\right] \;=\; \int f\left(\frac{hp(h)}{N_0}\right) dm(h). \tag{4}$$

Optimizing a point to point wireless communication channel entails finding the resource allocation function $p(h)$ that maximizes the metric c in (4). Maximizing (4) without further constraints is simple. Assuming that f is monotonically increasing, which is just a way of saying that more transmit power results in better communication quality, the solution is to transmit as much power as possible. This is detrimental to battery life, however. We therefore consider the average power consumption as well and formulate the following optimization problem

$$\begin{aligned} p^*(h) \;=\; & \operatorname*{argmin}_{p(h)} \; \mathbb{E}\left[f\left(\frac{hp(h)}{N_0}\right)\right] \\ & \text{s.t.} \qquad \mathbb{E}\left[p(h)\right] \leq p. \end{aligned} \tag{5}$$

This formulation attempts to distribute a long term power budget p over instantaneous fading realizations in order to maximize a long term performance metric. Observe how this simplest of wireless communication problems has led us to a functional optimization. Our goal is to find the *function* $p^*(h)$ that maps fading realizations $h(t)$ to power allocations $p(t) = p(h(t))$ that maximize c subject to not exceeding the power budget p. This is not that difficult to solve for point to point channels. But the moment we consider networks, as we do in Section 2, the resulting *functional* optimization problems become intractable.

In introducing (5) we encounter the three components of a wireless communication channel: (i) A probability distribution $m(h)$ that determines channel realizations h. (ii) A resource allocation function $p(h)$ that maps channel observations to selected transmit powers. This is the variable we want to determine. (iii) A function f that maps SNR values to instantaneous performance. In the first part of this exercise we address the modeling of the fading distribution $m(h)$ and the modeling of the performance function f. We are not interested in determining optimal power allocations $p(t)$. We will work on the more challenging versions of (5) that are formulated in network settings; see Section 2.

1.1 Fading Distributions

The fading distribution $m(h)$ characterizes the distribution of the channel realization h. In the modeling of wireless channels, we separate the modeling of the average of h and the modeling of the distribution itself.

1.2 Pathloss

The pathloss of a wireless channel is its expectation $\mathbb{E}(h)$. It characterizes the reduction in power of the transmitted signal as it propagates to the receiver. Pathloss values result from many effects. Free-space loss, refraction, diffraction, reflection and absorption. A simple, yet not inaccurate model, is to write the channel's expected value as

$$\mathbb{E}(h) = \Big(d_0 \,/\, d \Big)^{\gamma}, \tag{6}$$

where d_0 is a reference distance and $\gamma \in [2, 4]$ is a loss exponent. Both of these are empirically determined by fitting measurements in specific environments. More complex models are available for specific environments. But the model in (6) suffices for our illustrative examples.

1.1 Channel class. Create a channel class to evaluate pathloss. The attributes of the class are the reference distance d_0 and the pathloss exponent γ. Add a method to compute the pathloss as per (6) when given a distance d.

1.2 Testing. Compute and plot $\mathbb{E}(h)$ for d varying between $1m$ and $100m$. Channels are often more interesting if shown in a logarithmic scale. Set d_0 and γ to the values shown in the table in Figure 1.

1.3 Fading

Although we are using fading to refer to the channel distribution, it is more common to use the term fading to refer to the random channel variations around the mean. Formally, consider a random variable $\tilde{h}$ and write channel realizations as

$$h = \mathbb{E}(h) \times \Big(\tilde{h} \,/\, \mathbb{E}(\tilde{h}) \Big). \tag{7}$$

The random variations that $\tilde{h}$ describes are the result of the interference of the propagation wave with its own reflections and diffractions. A common model for this random variable is to posit an exponential distribution. Thus, the variable $\tilde{h}$ has a probability density

$$p_{\tilde{h}}(\tilde{h}) = (1/s)\, e^{-\tilde{h}/s}. \tag{8}$$

In (8) the constant s determines the amount of fading variability. Increasing s results in more variability in channel realizations. Although it is characterized by an exponential distribution, the fading model in (8) is called a Rayleigh fading model. This is because $\tilde{h}$ represents an energy. The channel is the squared root of this energy. Which has a Rayleigh distribution.

Parameter	Symbol	Value
Pathloss reference distance	d_0	1 m
Pathloss exponent	γ	2.2
Fading energy	s	2
Noise floor	N_0	10^{-6} mW
User density	ρ	0.05 users/m^2
Receiver maximum distance	w_c	50 m

Figure 1. Parameter values shared by all lab questions. Some other parameters may change from question to question and are specified as needed.

1.3 Fading Channel Class. Modify the class you created in Section 1.2 so that it incorporates s as an attribute. Write a method that returns Q random samples of the channel h that follow the model in (7) and (8).

1.4 Testing. Consider distances varying between $1m$ and $100m$ For each distance evaluate $Q = 100$ channel realizations. Plot the range of channels that you observe for each distance. You should see that there is always a sizable change of drawing a channel that is close to $h = 0$. This is why this phenomenon is called fading. The channel has a tendency to fade away at random. This is also the reason why your wireless devices offer a poor user experience. Set d_0, γ, and s to the values shown in the table in Figure 1.

1.4 Channel Performance

The performance function in (2) may refer to a variety of metrics. It can measure error rates, delay, or channel rates. We work here with channel rates and we assume the use of capacity achieving codes so that we can write

$$c(h) = \log\left(1 + \frac{hp}{N_0}\right). \tag{9}$$

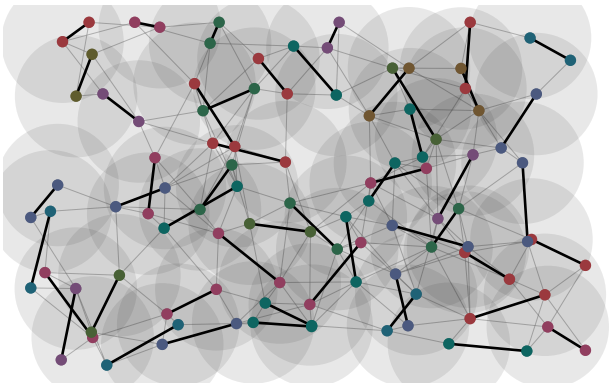

Figure 2. Representation of a wireless communication network. The links between i and $j = r(i)$ are represented by thick lines and the connections between i and other possible receivers $j \neq r(i)$ are denoted by thin lines. We also signify an interference range that represents the area where significant power from a transmitter is present in space.

1.5 Channel capacity method. Enhance the channel class of Section 1.3 with a method that takes as input a vector with Q power values p and Q fading channels h and returns channel capacity values that follow the model in (9). You can make this a separate class if you prefer. In either case, the noise floor power N_0 is an attribute of the class.

1.6 Testing. Consider distances varying between $1m$ and $100m$. For each distance evaluate $Q = 100$ channel realizations. Plot the range of channel capacities that you observe for each distance when the transmit power is $p = 0.05\ mW$ for all of the Q fading realizations. Set d_0, γ, s, and N_0, to the values shown in the table in Figure 1.

2 Wireless Networks

A wireless network is made up of several communicating pairs that share the same piece of the electromagnetic spectrum. Specifically, we consider a wireless system made up of a set of n transmitters and n receivers. Each transmitter $i \in \{1, 2, ..., n\}$ is paired with a single receiver

$r(i) \in \{1, 2, ..., n\}$. Although not required we have in mind large scale wireless systems with n ranging from several tens to several hundreds.

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. The notation here can be a little misleading, The channel $h_{ii}(t)$ connects transmitter i to its receiver $r(i)$, not to i itself. Likewise, the channel $h_{ij}(t)$ connects i to the receiver $r(j)$ of transmitter j, not to transmitter j itself. Figure 2 is a schematic representation of this network. The links between i and $r(i)$ are represented by thick lines and the connections between i and other possible receivers $r(j) \neq r(i)$ are denoted by thin lines. We also signify an interference range that represents the area where significant power from a transmitter is present in space.

The most important difference between the point to point wireless communication channels of Section 1 and the wireless networks we study in this section is not the presence of multiple transceiver pairs but the fact that wireless networks are limited by interference, not noise. The quantity that is determinant in the quality of a communication link is not the SNR as in (2) but the signal to interference plus noise ratio (SINR). This is because the signals of transmitters $j \neq i$ are perceived as noise by receiver $r(i)$. Thus, the function that maps transmit powers and channel strengths to link performances takes the form in (2) but with interference added to the noise level N_0.

For a more concrete mathematical model let $p_i(t)$ be the power that is transmitted by agent i at time t. The power that makes it to the intended receiver $r(i)$ is $h_{ii}(t)p_i(t)$ because $h_{ii}(t)$ denotes the channel connecting i to its intended receiver. But the transmissions of other agents also make it to $r(i)$. They do so with power $h_{ji}(t)p_j(t)$ because transmitter j transmits with power $p_j(t)$ and the channel from j to $r(i)$ is $h_{ji}(t)$. Thus, the communication performance $c_i(t)$, experienced by the ith transceiver pair can be written as

$$c_i(t) = f\left(\frac{h_{ii}(t)p_i(t)}{N_0 + \sum_{j\neq i} h_{ji}p_j(t)}\right). \tag{10}$$

In (10), the product $h_{ii}(t)p_i(t)$ is the received signal power, N_0 is the noise power and $\sum_{j\neq i} h_{ji}p_j(t)$ is the interference power.

We can now build from (10) towards an optimization problem that specifies the optimal allocation of resources in a wireless network. This is akin to what we did when building from(2) towards (5).

Start the process by arranging all of the transmit powers into the vector $\mathbf{p}(t) = [p_1(t); \ldots; p_n(t)]$ and all of the channels in the interference graph $\mathbf{H}(t) \in \mathbb{R}^{n \times n}$ with entries $[\mathbf{H}(t)]_{ij} = h_{ij}(t)$. The latter is a random adjacency matrix whose entries are independent of each other and follow the models of Section 1.1. Given a random realization $\mathbf{H}$, our goal is to identify a power allocation function $\mathbf{p}(\mathbf{H})$ so that whenever we observe the interference graph $\mathbf{H}(t)$ we allocate powers $\mathbf{p}(t) = \mathbf{p}(\mathbf{H}(t))$. To find a suitable function $\mathbf{p}(\mathbf{H})$ we observe that the long term performance of the ith communication channel can be written as

$$c_i = \mathbb{E}\left[f\left(\frac{h_{ii} p_i(\mathbf{H})}{N_0 + \sum_{j \neq i} h_{ji} p_j(\mathbf{H})} \right) \right] \tag{11}$$

This is analogous to the expression in (4) except that the performance is c_i because we are looking at channel i. And, more importantly, the factor that determines channel quality is not the SNR as in (4) but the SINR. These distinctions come from the difference between (2) and (10). An aspect of (11) that we emphasize is that the power allocation choice of each transmitter is a function of the whole interference graph. The power allocations are of the form $p_i(\mathbf{H})$ for all i.

To simplify upcoming discussions we define the the function $f_i\big(\mathbf{p}(\mathbf{H}), \mathbf{H}\big)$ as

$$f_i\big(\mathbf{p}(\mathbf{H}), \mathbf{H}\big) := f\left(\frac{h_{ii} p_i(\mathbf{H})}{N_0 + \sum_{j \neq i} h_{ji} p_j(\mathbf{H})} \right), \tag{12}$$

to represent the mapping of the interference graph $\mathbf{H}$ and the resource allocation $\mathbf{p}(\mathbf{h})$ into the instantaneous performance metric of channel i.

With this notational convention we further consider long term power consumptions $\mathbb{E}[p_i(\mathbf{H})]$ to write down the optimization problem

$$\begin{aligned} \mathbf{p}^*(\mathbf{H}) \;=\; & \underset{\mathbf{p}(\mathbf{H})}{\operatorname{argmax}} \; \sum_{i=1}^{n} \mathbb{E}\left[f_i\big(\mathbf{p}(\mathbf{H}), \mathbf{H}\big) \right], \\ & \text{s.t.} \qquad p_i \geq \mathbb{E}\left[p(\mathbf{H}) \right]. \end{aligned} \tag{13}$$

The problem in (13) models a wireless network in which we measure the interference graph $\mathbf{H}$. Based on this measurement we allocate powers $\mathbf{p}(\mathbf{H})$. These powers are such that the long term power consumption $\mathbb{E}[p(\mathbf{H})]$ does not exceed p_i and they result in long term performances c_i as stated in (11). We search for the resource allocation $\mathbf{p}^*(\mathbf{H})$ that maximizes the sum of the long term performances $\mathbb{E}[f_i(\mathbf{p}(\mathbf{H}), \mathbf{H})]$.

The optimization problem in (13) is very challenging to solve. Over the years, several heuristics have been developed to approximate its solution. Our approach here, which we pursue in Section 3 is to use a *learned* heuristic. Before doing that, we work on building the pieces of the model that we will use in the development of this heuristic.

2.1 Wireless Network Simulation

The channel coefficient between transmitter i and receiver $r(j)$ can still be seen as a random variable drawn from a fading distribution as in Section 1.1. Similarly, we can model channel performance using the rate functions of Section 1.4. The difference is that we now need to consider multiple channels and interference. We will base the network simulation in a spatial distribution of agents.

We consider a scenario in which n transmitters are dropped in an area of dimensions w_x by w_y. The transmitters are dropped uniformly at random. An important parameter of the network is the transmitter density, defined as

$$\rho = (w_x w_y) \,/\, n. \tag{14}$$

In our network model each transmitter i is associated with receiver $r(i)$. Receiver $r(i)$ is dropped uniformly at random within a circle of radius w_c center at the position of transmitter i.

2.1 Transmitter and Receiver positioning.. Write a class whose attributes are the the dimensions w_x, w_y, and w_c along with the number of nodes n. These numbers are provided at initialization. We add attributes to represent the positions of transmitters and receivers. These are generated at initialization with transmitters and receivers placed at random as described above. Test with $w_x = 200\ m$ and $w_y = 100\ m$. Set the user density ρ and the radius w_c to the values shown in the table in Figure 1.

2.2 Pahtloss. Enhance the network class to include the generation of a pathloss matrix $\mathbb{E}(\mathbf{H})$. The entries $\mathbb{E}(h_{ij})$ of $\mathbb{E}(\mathbf{H})$ follow the model in (1.2). Doing so requires that we add the reference distance d_0 and the pathloss exponent γ as attributes of the class. Different from Section 1.2,

$\mathbb{E}(\mathbf{H})$ is stored as an attribute of the class. Test the class with the parameter values shown in the table in Figure 1. Use $w_x = 200\ m$ and $w_y = 100\ m$.

2.3 Fading. Add the fading parameter s as an attribute and a method that takes as input a number Q and generates Q samples of the interference graph $\mathbf{H}$. The entries of $\mathbf{H}$ follow the model in (1.3). Test the class with the parameter values shown in the table in Figure 1. Use $w_x = 200\ m$ and $w_y = 100\ m$ and $Q = 100$.

2.4 Channel Capacity. Create a function that takes as input Q power allocation vectors $\mathbf{p}$ and Q realizations of the interference graph $\mathbf{H}$ and returns the capacities c_i for all of the agents. These capacities follow the model,

$$c_i(\mathbf{p}, \mathbf{H}) = \log\left(1 + \frac{h_{ii}p_i}{N_0 + \sum_{j\neq i} h_{ji}p_j}\right). \tag{15}$$

Test with the parameter values shown in the table in Figure 1. Use $w_x = 200m$, $w_y = 100m$ and $Q = 100$.

3 Random Edge Graph Neural Networks

We solve (13) with a learning parametrization. More specifically, we propose to use a graph neural network (GNN) to map channel graph realizations $\mathbf{H}$ into power allocations $\mathbf{p}(\mathbf{H})$. To emphasize that the input to the GNN is a graph with edges that come from a random distribution, we rename the GNN a random edge (RE) GNN.

As any other GNN, the REGNN composes L layers, each of which is itself the composition of a graph filter with a pointwise nonlinearity. Introduce then a layer index l and a set of 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 $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ to produce the output of the lth layer

as

$$\mathbf{x}_l = \sigma\big[\mathbf{y}_l\big] = \sigma\Big[\mathbf{A}_l(\mathbf{H})\mathbf{x}_{l-1}\Big] = \sigma\left[\sum_{k=0}^{K_l-1} \alpha_{lk}\mathbf{H}^k\mathbf{x}_{l-1}\right]. \tag{16}$$

A REGNN is defined by recursive application of (17). The input to this recursion is a node state vector $\mathbf{x}_0 = \mathbf{x}$. The output is the Lth layer signal $\mathbf{x}_L$. Observe that in a REGNN the real input is the interference graph $\mathbf{H}$. The node state vector $\mathbf{x}_0$ could become useful for incorporating node properties. But in this lab it suffices to think of it as an arbitrary vector.

To increase the expressive power of REGNNs we consider multiple features per layer. At Layer l we consider a total of F_l features, each of which is a graph signal $\mathbf{x}_l^f$. These features are grouped in the $n \times F_l$ matrix graph signal $\mathbf{X}_l$ in which each of the columns represents a different graph signal. To process these signals with multiple features we resort to the use of multiple input multiple output (MIMO) graph filters. At layer l the input to the MIMO filter is the $n \times F_{l-1}$ matrix graph signal $\mathbf{X}_{l-1}$ and the output is the $n \times F_l$ matrix graph signal $\mathbf{Y}_l$. The latter is defined according to the relationship

$$\mathbf{Y}_l = \sum_{k=0}^{K-1} \mathbf{H}^k\mathbf{X}_{l-1}\mathbf{A}_{lk} \tag{17}$$

where the matrices $\mathbf{A}_{lk}$ are collections of $F_{l-1} \times F_l$ filter coefficients α_{lk}^{fg}. The coefficients α_{lk}^{fg} represent a regular graph filter that processes the fth column of $\mathbf{X}_{l-1}$ to produce the gth column of $\mathbf{Y}_l$. Each layer of a REGNN with multiple features processes (17) with a pointwise nonlinearity, yielding the recursive definition

$$\mathbf{X}_l = \sigma\left[\sum_{k=0}^{K-1} \mathbf{H}^k\mathbf{X}_{l-1}\mathbf{A}_{lk}\right] \tag{18}$$

A REGNN with multiple features is defined by recursive application of (18). The input to this recursion is the node state signal $\mathbf{X}_0 = \mathbf{X}$. The output is the lth layer (single feature) signal $\mathbf{Z}_L := \mathbf{z}_L$. For future reference we group all filter coefficients in the filter tensor $\mathbf{A} = \{A_k\}_k$ and define the REGNN operator as,

$$\Phi(\mathbf{H}, \mathbf{A}) = \mathbf{X}_L. \tag{19}$$

The operator in (19) is a graph neural network 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 Φ.

Our goal is to solve (13) using as inputs the class of functions that can be represented by the REGNNs in (19). This translates to solving the optimization problem

$$\begin{aligned} \mathbf{A}^* \;=\; & \underset{\mathbf{A}}{\operatorname{argmax}} \; \sum_{i=1}^{n} \mathbb{E}\left[f_i\big(\Phi(\mathbf{H},\mathbf{A}),\mathbf{H}\big)\right], \\ & \text{s.t.} \qquad p_i \geq \mathbb{E}\left[\Phi(\mathbf{H},\mathbf{A})\right]. \end{aligned} \tag{20}$$

The REGNN receives as input a random realization of the underlying graph shift operator $\mathbf{H} \sim m(\mathbf{H})$. According to (20), the filter coefficients in the tensor $\mathbf{A}$ are trained relative to the statistics of the interference graph $\mathbf{H}$.

3.1 A First Approach to Training

There are several complications to solving (20). The first one is that we are considering a constrained optimization problem. This is different from the unconstrained ERM problems we have considered so far. To overcome this roadblock we formulate a penalized version of (20). To do so, let. $\mu_i > 0$ be importance weights associated with the different constraints in (20) and define the optimization problem,

$$\mathbf{A}^* \;=\; \underset{\mathbf{A}}{\operatorname{argmax}} \; \sum_{i=1}^{n} \mathbb{E}\left[f_i\big(\Phi(\mathbf{H},\mathbf{A}),\mathbf{H}\big)\right] - \mu_i \mathbb{E}\left[\Phi(\mathbf{H},\mathbf{A})\right]. \tag{21}$$

Solving (21) is not the same as solving (20). Among many other differences, there is no guarantee that the power constraints are satisfied. However, it does gives us a problem that we know how to solve. The problem in (21) is an ERM problem. The loss function has a complicated expression. It involves link performance functions and weighted power consumptions summed over all agents. But it is still just a loss function.

3.1 Unconstrained training. Generate a network object using the class you created in Section 2.1. Use the parameters in the table in Figure 1 with $w_x = 200$ m, $w_y = 100$ m and $\mu_i = 0.01$. Train a GNN with 200 iterations

and batch size set to 100 to solve (21). We recommend that you use $L = 5$ layers with $K_l = 5$ taps and $F_l = 1$ features per layer, and a step size of 0.01. Use $\mathbf{x} = \mathbf{1}$ as the input signal. Recall that the network channels $\mathbf{H}$ are drawn randomly. Test the loss with 10^4 channel realizations.

3.2 Stability. Our stability analyses indicate that the GNN we trained in Question 3.1 should work in other networks. Generate 1000 other networks with the same parameters of Question 3.1 and evaluate the mean and variance of the GNN's performance when executed in this family of networks. In your loss evaluations use 10^4 channel realizations.

3.3 Transference. Our experience with transferability indicates that the GNN we trained in Question 3.1 should work in other networks with different numbers of nodes. Generate 1000 networks with $w_x = 400m$, $w_y = 200m$ and evaluate the mean and variance of the GNN's performance when executed in this family of networks. In your loss evaluations use 10^4 channel realizations.

4 Learning with Constraints

To find the optimal filter tensor $\mathbf{A}^*$ in the constrained optimization problem in (20) we find a saddle point of its Lagrangian. To define the Lagrangian let $\boldsymbol{\mu} \geq \mathbf{0}$ collect multipliers μ_i associated each of the power constraints in (20). The Lagrangian of (20) is the following weighted combination of objective and constraints

$$\mathcal{L}(\mathbf{A}, \boldsymbol{\mu}) \;=\; \sum_{i=1}^{n} \mathbb{E}\left[f_i\big(\Phi(\mathbf{H}, \mathbf{A}), \mathbf{H}\big)\right] - \mu_i(\mathbb{E}\left[\Phi(\mathbf{H}, \mathbf{A})\right] - p_i). \tag{22}$$

Observe that (22) is the argument that appears in (20). This is not coincidental. We introduced (20) as an intermediate step towards learning with constraints.

A saddle point of the Lagrangian in (22) is a primal dual pair $(\mathbf{A}^\dagger, \boldsymbol{\mu}^\dagger)$ such that for all variables $\mathbf{A}$, and $\boldsymbol{\mu}$ in a sufficiently small neighborhood

we have

$$\mathcal{L}\Big(\mathbf{A}, \boldsymbol{\mu}^\dagger\Big) \leq \mathcal{L}\Big(\mathbf{A}^\dagger, \boldsymbol{\mu}^\dagger\Big) \leq \mathcal{L}\Big(\mathbf{A}^\dagger, \boldsymbol{\mu}\Big). \tag{23}$$

Namely, the saddle point is locally maximal on the primal variables and locally minimal in the dual variables. There several saddle points satisfying (23) because the Lagrangian $\mathcal{L}(\mathbf{A}, \boldsymbol{\mu})$ in (22) is associated to a nonconvex optimization problem. 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 $\boldsymbol{\mu}$ with gradient ascent steps in the primal variables $\mathbf{A}$. Taking gradients with respect to the filter tensor $\mathbf{A}$ and the multiplier $\boldsymbol{\mu}$ is challenging because of the expectation operators. The gradient descent update with respect to $\boldsymbol{\mu}$, for example, takes the form

$$\mu_{i,k+1} = \mu_{i,k} - \epsilon_1 \left(\mathbb{E}\left[\Phi(\mathbf{H}, \mathbf{A}_k)\right] - p_i\right). \tag{24}$$

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

$$\mu_{i,k+1} = \mu_{i,k} - \epsilon_1 \left(\Phi(\mathbf{H}_k, \mathbf{A}_k) - p_i\right). \tag{25}$$

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

$$\mathbf{A}_{k+1} = \mathbf{A}_k + \epsilon_2 \left[\sum_{i=1}^{n} \nabla_{\mathbf{A}} f_i\big(\Phi(\mathbf{H}, \mathbf{A}), \mathbf{H}\big) - \mu_i \nabla_{\mathbf{A}} \Phi(\mathbf{H}, \mathbf{A})\right]. \tag{26}$$

Equations (25) and (26) specify a stochastic primal-dual method for solving the constrained optimization problem in (20).

4.1 Constrained training. Generate a network object using the class you created in Section 2.1. Use the parameters in the table in Figure 1 with $w_x = 200m$ and $w_y = 100m$. Train a GNN to solve (20) using the primal dual method specified by (25) and (26). Set the power budget to $p_i = 10^{-3}mW$ for all i. We recommend that you use $L = 5$ layers with $K_l = 5$ taps and $F_l = 1$ feature per layer. $\boldsymbol{\mu}$ is initialized as $[0, .., 0]$ and

the step size for the primal is $\epsilon_2 = 0.01$ and the dual $\epsilon_1 = 0.001$. Use $\mathbf{x} = \mathbf{1}$ as the input signal. Recall that the network channels $\mathbf{H}$ are drawn randomly. Observe that since we are now looking at a constrained optimization problem, evaluating the loss entails evaluating the objective and evaluating the constraints. Thus, in your testing, you need to evaluate the loss and you also need to evaluate the average power constraints. In your loss evaluations use 10^4 channel realizations.

4.2 Stability. Our stability analyses indicate that the GNN we trained in Question 4.1 should work in other networks. Generate 1000 other networks with the same parameters of Question 4.1 and evaluate the mean and variance of the GNN's performance when executed in this family of networks. You need to evaluate the cost and the constraints. In your loss evaluations use 10^4 channel realizations.

4.3 Transference. Our experience with transferability indicates that the GNN we trained in Question 4.1 should work in other networks with different numbers of nodes. Generate 1000 networks with $w_x = 400m$, $w_y = 200m$ and evaluate the mean and variance of the GNN's performance when executed in this family of networks. You need to evaluate the cost and the constraints. In your loss evaluations use 10^4 channel realizations.

4.1 Model Free Learning

It is germane to point out that to implement (25) 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 $\mathbf{H}_k$ and implement the resource allocation $\mathbf{\Phi}(\mathbf{H}_k, \mathbf{A}_k)$ according to the current filter tensor iterate $\mathbf{A}_k$. We can then observe the reward outcome $f_i(\mathbf{\Phi}(\mathbf{H}_k, \mathbf{A}_k), \mathbf{H}_k)$ and use it to implement the Lagrange multiplier update in (25).

The same is not quite true for the update in (26) 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. This is an interesting twist but we are not going to study it in this lab.

5 Report

Do not take much time to prepare a lab report. We do not want you to report your code and we don't want you to report your work. Just give us answers to questions we ask. Specifically give us the following:

Question	**Report deliverable**
Question 1.1	Do not report.
Question 1.2	Plot.
Question 1.3	Do not report.
Question 1.4	Plot.
Question 1.5	Do not report.
Question 1.6	Plot.
Question 2.1	Network Plot.
Question 2.2	Do not report.
Question 2.3	Do not report.
Question 2.4	Do not report.
Question 3.1	Average loss over 10^4 channel realizations. Average capacities per node. Average powers per node.
Question 3.2	Average loss for 10^3 networks estimated with 10^4 channels each. Loss variance across 10^3 networks estimated with 10^4 channels each.
Question 3.3	Average loss for 10^3 networks estimated with 10^4 channels each. Loss variance across 10^3 networks estimated with 10^4 channels each.

Question	Report deliverable
Question 4.1	Average loss over 10^4 channel realizations. Average capacities per node. Average powers per node.
Question 4.2	Average loss for 10^3 networks estimated with 10^4 channels each. Loss variance across 10^3 networks estimated with 10^4 channels each. Average and variance of long term power consumption
Question 4.3	Average loss for 10^3 networks estimated with 10^4 channels each. Loss variance across 10^3 networks estimated with 10^4 channels each. Average and variance of long term power consumption

We will check that your answers are correct. If they are not, we will get back to you and ask you to correct them. As long as you submit responses, you get an A for the assignment. It counts for 16% of your final grade.