Resource Allocation in Large-Scale Wireless Control Systems with Graph Neural Networks ⋆

Vinicius Lima * **Mark Eisen** ** **Konstatinos Gatsis** ***
Alejandro Ribeiro *

* *Electrical and Systems Engineering, University of Pennsylvania Philadelphia, PA 19104 USA (e-mail: {vlima, aribeiro}@seas.upenn.edu).*
** *Intel Corporation, Hillsboro, OR 80523 USA (e-mail: mark.eisen@intel.com).*
*** *Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, UK (e-mail: konstantinos.gatsis@eng.ox.ac.uk).*

Abstract: Modern control systems routinely employ wireless networks to exchange information between a large number of plants, actuators and sensors. While wireless networks are defined by random, rapidly changing conditions that challenge common control design assumptions, properly allocating communication resources helps to maintain operation reliable. Designing resource allocation policies is usually challenging and requires explicit knowledge of the system and communication dynamics, but recent works have successfully explored deep reinforcement learning techniques to find optimal model-free resource allocation policies. Deep reinforcement learning algorithms do not necessarily scale well, however, which limits the immediate generalization of those approaches to large-scale wireless control systems. In this paper we discuss the use of reinforcement learning and graph neural networks (GNNs) to design model-free, scalable resource allocation policies. On the one hand, GNNs generalize the spatial-temporal convolutions present in convolutional neural networks (CNNs) to data defined over arbitrary graphs. In doing so, GNNs manage to exploit local regular structure encoded in graphs to reduce the dimensionality of the learning space. The architecture of the wireless network, on the other, defines an underlying communication graph that can be used as basis for a GNN model. Numerical experiments show the learned policies outperform baseline resource allocation solutions.

Keywords: Resource Allocation; Control over networks; Graph Neural Networks; Reinforcement Learning Control; Neural Networks

1. INTRODUCTION

Modern control systems often use wireless communication networks to exchange information between plants, sensors and actuators. Wireless networks add flexibility to the control system, but also make the design of control and communication policies more challenging (Hespanha et al., 2007; Park et al., 2018). Wireless networks are characterized by rapidly changing, random communication conditions known as fading. The reliability of the communication loop depends not only on the fading state of the communication channel but also on the resources assigned to that particular channel. Communication resources are usually limited, however, and it is natural to look for an optimal way to distribute the resources available in the network. Mathematically, such resource allocation problems involve optimizing some performance metric over an allocation function, which usually results in an infinite dimensional problem, cf. e.g. (Cao and Li, 2001; Eryilmaz and Srikant, 2007). The formulation of the resource allocation problem resembles that of a statistical learning problem, allowing one to leverage machine learning techniques for resource allocation (Eisen et al., 2019; Liang et al., 2018). In the case of wireless control systems, we need to balance metrics such as power consumption, latency and reliability while assuring proper operation of the control plants. That is precisely the problem works on resource allocation and scheduling tackle, cf. e.g. (Rehbinder and Sanfridson, 2004; Shi et al., 2011; Gatsis et al., 2015; Charalambous et al., 2017).

The dynamic nature of the problem points to reinforcement learning as a possible framework to solve it. Reinforcement learning aims to represent the notion that learning occurs in interaction with the environment and its structure is particularly suitable for problems where explicit information about underlying models is limited. The association of reinforcement learning with deep neural networks, in particular, recently led to impressive results in applications such as AlphaGO (Silver et al., 2016). Those results in turn motivated the use of similar deep reinforcement learning techniques in other areas, among them resource allocation for wireless control systems, cf. (Demirel et al., 2018; Leong et al., 2018; Baumann et al., 2018) and our previous work (Lima et al., 2020).

Deep reinforcement learning algorithms, however, usually suffer from sample complexity. As the dimensionality of the problem grows, so do the number of parameters to be learned and the number of samples necessary to train the algorithm. The dimensionality of the learning space, however, can be reduced if we are able to exploit some symmetry or particular structure of the problem. In wireless networks and wireless control systems, we can turn to the underlying communication graph to param-

⋆ Supported by Intel Science and Technology Center for Wireless Autonomous Systems and ARL DCIST CRA W9111NF-17-2-0181.

eterize the allocation policy (Lee et al., 2019); in particular, we substitute graph neural networks (GNNs) instead of traditional neural networks (Eisen and Ribeiro, 2019; Shen et al., 2019). GNNs generalize time- or spatial-domain convolutions in traditional convolutional neural networks (CNNs) to explore regularity in general graph structures (Scarselli et al., 2009; Gama et al., 2019). Training in GNNs boils down to learning the coefficients of the filters used to aggregate information from the network nodes — instead of the weights of the linear combinations at each hidden unit as in multilayer neural networks —, thus reducing the number of overall parameters the algorithm must learn.

Here we consider a centralized setting where independent control plants share a wireless communication medium. Allocation decisions are taken by a remote base station (BS) based on estimates of current plant states and channel conditions (Section 2). The wireless communication model takes into account fading and interference, defining a graph that allows us to use GNNs (Section 4) to parameterize the resource allocation policy. A standard reinforcement learning framework (Section 3) is then used to learn the coefficients (i.e. graph filters) of the resource allocation policy. Numerical experiments (Section 5) illustrate the use of the proposed approach. Throughout the paper uppercase letters refer to matrices and lowercase letters to vectors. Positive (semi)definiteness of a matrix is indicated by $X(\geq) > 0$. $\mathbb{R}$ and $\mathbb{N}$ stand for the set of real and natural numbers.

2. RESOURCE ALLOCATION IN CONTROL SYSTEMS

Consider a system made up by m independent plants sharing a common wireless medium as exemplified in Figure 1. As is common in large scale applications — e.g. industrial control — the plants are organized into various cells; within each cell the plants share a common wireless base station (BS). We further assume each BS is configured in a multiple-input-single-output (MISO) configuration, with dedicated transmit antennas serving each of the plants in its cell. At each time instant a plant samples its state and sends this information to its base station (BS) containing a centralized controller. The base station then sends the corresponding control signals back to the plants with some allocated resources based on the plant states and channel conditions. The dynamics of each plant i is given by a discrete, time-invariant but not necessarily linear model $f(\cdot,\cdot) : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^p$ mapping a current state vector $x_t^{(i)} \in \mathbb{R}^p$ and corresponding control input $u_t^{(i)} \in \mathbb{R}^q$ to the next state of the system. Each of those plants is affected by some random noise $w_t^{(i)} \in \mathbb{R}^p$ with covariance matrix $W^{(i)} \in \mathbb{R}^{p\times p}$ standing for eventual disturbances and unmodeled dynamics,

$$x_{t+1}^{(i)} = f\left(x_t^{(i)}, u_t^{(i)}\right) + w_t^{(i)}. \tag{1}$$

The control input $u_t^{(i)}$ is designed as some function of the state, i.e. $u_t^{(i)} = g(x_t^{(i)})$, such as a standard linear quadratic regulator in the case of linear plants.

In this setting the plants close their feedback loop over a shared wireless network overseen by a base station. This medium is inherently noisy and prone to packet drops. When the plant is able to successfully receive the control signal, the feedback loop is closed, and the plant can execute the ensuing control action. When the plant cannot reliably receive the signal, however, it relies on a local estimate of the current control signal.

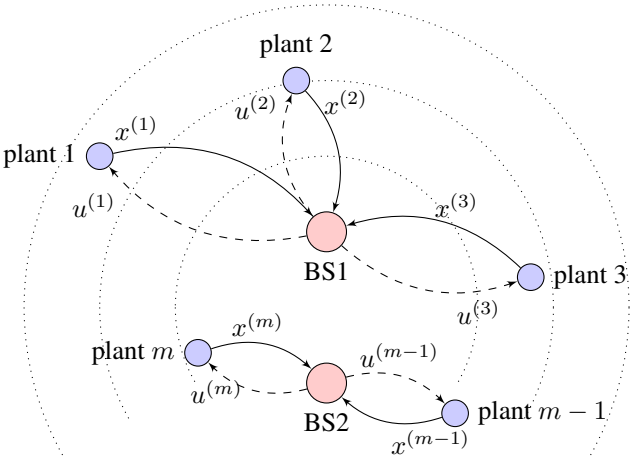

Fig. 1. Wireless control system made up by m independent plants with states $x^{(i)}$.

Under these assumptions, the dynamics of each plant can be represented by

$$x_{t+1}^{(i)} = \begin{cases} f\left(x_t^{(i)}, u_t^{(i)}\right) + w_t^{(i)}, & \text{closed loop,} \\ f\left(x_t^{(i)}, \tilde{u}_t^{(i)}\right) + w_t^{(i)}, & \text{open loop,} \end{cases} \tag{2}$$

where $\tilde{u}_t^{(i)}$ is an estimate of $u_t^{(i)}$ computed locally at the plant, such as the last successfully received control signal.

The probability of closing the feedback loop over the communication channel will depend on the resources or power assigned to the plant as well as a random channel state known as wireless fading. Let then $h^{(ii)}$, $i \in [m]$, a random variable describing the fading state experienced between the plant i and its paired antenna in its cell's BS. That fading state is comprised of a constant slow fading term dependent on the distance between the plant and the BS, and a random fast fading term drawn from a probability distribution $o(h)$. Moreover, since multiple control plants share the communication network, information packets from a plant might interfere with those from others. That interference is described by a random variable $h^{(ji)}$, which reflects the fading state between plant j's dedicated transmit antenna and plant i . These can be aggregated in a random matrix H describing the fading states and interference coefficients of the communication model,

$$H = \begin{bmatrix} h^{(11)} & \dots & h^{(1m)} \\ \vdots & \ddots & \vdots \\ h^{(m1)} & \dots & h^{(mm)} \end{bmatrix}. \tag{3}$$

Now, denote by $\alpha^{(i)} \in \mathbb{R}_+$ the power assigned to plant i. In this setting the signal to noise plus interference ratio for plant i will be given by

$$\text{SNIR}^{(i)} = \frac{h^{(ii)}\alpha^{(i)}(H)}{\sigma^2 + \sum_{j\neq i} h^{(ji)}\alpha^{(j)}(H)}, \tag{4}$$

with σ^2 the variance of the communication channel noise. The probability of the plant receiving the feedback control signal will depend on the ratio above. Formally, define a function $v : \mathbb{R}_+ \times \mathbb{R}_+ \to [0, 1]$ that, given a resource allocation distribution as well as fading and interference conditions, returns the probability of successfully receiving the information packet. We assume each control plant in the system will then close the feedback loop with probability

$$v(\alpha^{(i)}, H) = 1 - \exp\left(-\text{SNIR}^{(i)}\right). \tag{5}$$

This allows us to rewrite the system dynamics in (2) as

$$x_{t+1}^{(i)} = \begin{cases} f\left(x_t^{(i)}, u_t^{(i)}\right) + w_t^{(i)}, \text{ w.p. } v(\alpha^{(i)}, H), \\ f\left(x_t^{(i)}, \tilde{u}_t^{(i)}\right) + w_t^{(i)}, \text{ w.p. } 1 - v(\alpha^{(i)}, H). \end{cases} \tag{6}$$

From the structure of the control system (6) and equations (4) - (5), we can see that allocating more power to a plant will increase the chances of that plant closing its control loop. In most practical systems, however, we do not have unlimited power to allocate between the communication channels. The resource allocation problem consists of properly allocating resources available in the communication network while keeping all plants in desirable states. We want to find a resource allocation function $\alpha(H, x)$ that, given current fading and interference conditions aggregated in the interference matrix H_t (3) and plant states $x_t := [x_t^{(1)}; \ldots; x_t^{(m)}]$, distributes a maximum power budget p_{max} among the plants. It is important to keep the plants operating around an equilibrium point, and as such we consider a finite horizon, cumulative quadratic metric to assess the performance of the allocation function. The constrained resource allocation problem can then be formulated as

$$\begin{aligned} &\min_{\alpha(H,x)} \mathbb{E}_{x_0}^{\alpha(H,x)} \left[\sum_{t=0}^{T} x_t^\intercal Q_t x_t | x_0 = \hat{x}_0\right] \\ &\text{s. t. } \alpha(H, x) \in \mathcal{P}; \mathcal{P} = \left\{\alpha(H, x) \ : \ \sum_{i=1}^{m} \alpha^{(i)} \leq p_{\max}\right\}, \end{aligned} \tag{7}$$

with $Q_t \geq 0$ and $\alpha^{(i)}$ the ith component of the resource allocation vector $\alpha(H, x)$, i.e. resource allocated to plant i. At each time t, the BS uses power $\alpha_t^{(i)} = [\alpha(H_t, x_t)]_i$ to send the control signal $u^{(i)}$ back to plant i. The communication exchange subsequently occurs with success rate given by $v(\alpha_t^{(i)}, H_t)$ (5) and plant i evolves according to (6).

In (7), the objective involves finding the resource allocation function $\alpha(x, h)$ that results in the minimum operation cost of the plants while satisfying the resource constraints. Note that this leads to an (infinite-dimensional) optimization problem over a function $\alpha(x, h)$. It is generally intractable to find optimal solutions even for problems with a low number of plants and with short optimization horizons. Moreover, finding an optimal policy directly in (7) necessarily requires explicit knowledge of the plant dynamics and communication models in (6), which are often unavailable in practice. The challenging nature of the problem and the search for model-free allocation policies motivated recent works (Demirel et al., 2018; Baumann et al., 2018; Lima et al., 2020) to use deep reinforcement learning techniques to design resource allocation functions in wireless control systems.

3. REINFORCEMENT LEARNING FOR RESOURCE ALLOCATION

Reinforcement learning represents the idea that learning occurs in interaction with the environment: at each time step, an agent executes some action, observes the resulting state of the system, receives a cost from the environment and then tries to find actions that minimize the cumulative value of those one-step costs (Sutton and Barto, 2018). Reinforcement learning problems are mathematically described in terms of Markov Decision Processes (MDPs). A MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P} \rangle$ with $\mathcal{S}$ a set of states, $\mathcal{A}$ a set of actions and $\mathcal{P}$ a state transition probability kernel. The state transition probability kernel $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ assigns to each triplet (s, a, s') the probability of moving from state s to s' if action a is chosen. A transition from a state s_t to s_{t+1} incurs a cost per stage r_t, and the agent takes actions according to a stochastic policy $\pi(a|s)$ in a space Π. That policy corresponds to a probability distribution that gives the probability of the agent to choose an action a when its current state is s. The agent's objective is to minimize the cumulative cost

$$J(\pi, s) := \mathbb{E}_s^\pi [R_t] = \mathbb{E}_s^\pi \left[\sum_{k=1}^{T} \gamma^t r_{k+t+1}\right] \tag{8}$$

starting from state s and following a policy $\pi(\cdot)$ with $\gamma \in (0, 1]$ a given discount factor (Hernandez-Lerma and Lasserre, 1996; Sutton and Barto, 2018; Szepesvári, 2010). The optimization problem then consists in finding a policy π^* that achieves the minimum of the value function

$$J^*(s) = \inf_{\pi \in \Pi} J(\pi, s), s \in \mathcal{S}. \tag{9}$$

In the resource allocation setting we consider the base station as a centralized agent taking actions (allocation decisions) on $\mathbb{R}_+^m$ according to some resource allocation policy $p(H, x, u)$. At each time instant the base station defines the allocation policy based on estimates of the plant states and channel and interference conditions H, as well as on previously used control inputs. The state of the agent is then given by

$$s_t = [H_t; x_t] = \left[H_t; x_t^{(1)} \ldots; x_t^{(m)}; u_t^{(1)} \ldots; u_t^{(m)}\right].$$

As usual in control systems, the performance of the resource allocation function is measured by a quadratic cost that penalizes large deviations from the equilibrium point,

$$J(\pi, s) = \mathbb{E}_s^{p(h,x)} \left[\sum_{t=0}^{T} \gamma^t x_t^\intercal Q_t x_t | x_0 = \hat{x}_0\right]. \tag{10}$$

Here we will make use of policy-based reinforcement learning algorithms, and thus the resource allocation function $\alpha(h, x)$ is first parameterized with some stochastic policy $\pi(p|s; \theta)$. That policy is fully specified by some parameter vector $\theta \in \mathbb{R}^r$, i.e.

$$\alpha(h, x) = \pi(p|s; \theta). \tag{11}$$

The resource allocation problem can then be written as

$$\begin{aligned} &\min_{\theta} \mathbb{E}_{x_0}^{\pi(h,x;\theta)} \left[\sum_{t=0}^{T} \gamma^t x_t^\intercal Q_t x_t | x_0 = \hat{x}_0\right] \\ &\pi(h, x; \theta) \in \mathcal{P}; \mathcal{P} = \left\{\pi(h, x; \theta) \ : \ \sum_{i=1}^{m} \pi^{(i)} \leq p_{\max}\right\} \end{aligned} \tag{12}$$

Note that the cost function will now depend on the parameters θ, allowing us to take

$$J(\theta) = \mathbb{E}_{x_0}^{\pi(h,x;\theta)} \left[\sum_{t=0}^{T} \gamma^t x_t^\intercal Q_t x_t | x_0 = \hat{x}_0\right]. \tag{13}$$

To find the optimal allocation policy in this setting, at each iteration we will perform approximate gradient descent in the cost function $J(\theta)$ (Sutton and Barto, 2018, ch. 13). The gradient descent step is based on some estimate $\nabla J(\hat{\theta}_t)$ of the gradient of $J(\theta)$ with respect to θ, yielding

$$\theta_{t+1} = \theta_t - \beta \nabla J(\hat{\theta}_t), \tag{14}$$

where β is the step size or learning rate. This is the basic structure behind policy gradient methods. Note that the update rule in (14) does not take into account the underlying model of the agent or the environment — or, in our case, the dynamic model of the control plants and wireless communication network. The estimate $\nabla \hat{J}(\theta_t)$ in (14) is instead computed from the samples — i.e. actions, state transitions and costs — obtained during training. That is justified by the policy gradient theorem, according to which the gradient of the cost function, $\nabla J(\theta_t)$, is proportional to the gradient of the policy, $\nabla \pi(\cdot; \theta_t)$, and the expected return from following that policy. Given that relation, policy based methods then come up with strategies to sample actions, costs and states so as to approximate the gradient of the cost function. In particular, we will consider in this paper the classic REINFORCE algorithm, in which that estimate will depend on the action a_t taken at time t (Williams, 1992; Sutton et al., 2000),

$$\theta_{t+1} = \theta_t - \beta \gamma^t R_t \frac{\nabla \pi(a_t|s_t;\theta_t)}{\pi(a_t|s_t;\theta_t)}. \tag{15}$$

The equation shows that each update of the REINFORCE algorithm depends on the return R_t associated to the action a_t taken and the ratio between the gradient of the probability of executing that action and the probability of doing so (Sutton and Barto, 2018). Moreover, equations (11), (14) and (15) show that policy-based algorithms are suitable to applications involving continuous action spaces: policy parameterization (11) allows us to consider continuous policies directly, and the policy gradient theorem (14) — (15) gives a strategy to sample transitions from a continuous space.

There is some flexibility on the choice of the parameters θ, but in deep reinforcement learning, the parameters θ correspond to outputs of a multilayer neural network. This combination of deep learning with reinforcement learning led to impressive results in computer science (Mnih et al., 2015; Silver et al., 2016) and was later extended to other areas. Deep RL has also been successfully applied in wireless (Eisen et al., 2019; Liang et al., 2018) and wireless control systems (Demirel et al., 2018; Leong et al., 2018; Baumann et al., 2018; Lima et al., 2020). Deep reinforcement learning suffers from a sample complexity problem, however, which hinders the immediate generalization of those techniques to large-scale settings. Standard artificial neural networks consist of successive computations of linear combinations followed by nonlinear transformations. Deep RL algorithms must then learn the parameters used in the linear combinations at each hidden unit in the network. As the dimension of the input of the neural network — made up by the current state of the underlying MDP, associated actions and costs — grows, the number of parameters to be learned becomes prohibitively large. To overcome this dimensionality issue, we can look for some regular structure in the learning problem that reduces the dimensionality of the training space. In particular, note that the communication model in the wireless resource allocation problem (3) defines an underlying graph structure for the optimization problem. This suggests that we might leverage graph neural networks to parameterize the resource allocation function.

4. GRAPH NEURAL NETWORKS

We introduce the graph neural network (GNN) as an alternative to the standard, fully connected neural network to represent our resource allocation policy. GNNs can be viewed as a generalization to the popular convolutional neural network (CNN), a widely used deep learning tool in large scale applications such as image and speech recognition, in part by exploiting the regular structure of time and spatial data. In CNNs, the arbitrary linear operations used in standard deep neural networks (Bishop, 2006, ch. 5) is replaced with linear convolutional filters. This more controlled structure significantly reduces the number of overall parameters the model must learn during training, since the algorithm now learns the coefficients of the corresponding convolutional filters and not the weights of the linear combinations computed at every neuron in the network. Moreover, the dimensionality of the filters being learned is invariant to the size of the input data, making the CNN attractive for large scale applications. While the convolution employed by CNNs are naturally suited for processing of temporal or spatial data, the same does not hold true for inputs without such a regular structure. However, the wireless control system architecture considered here nonetheless contains structure embedded in the fading patterns H that can be incorporated into the policy parameterization—namely, this structure may be represented by a graph.

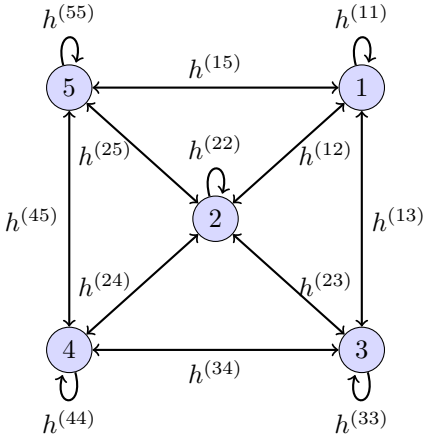

Fig. 2. Graph defined by the interference model (3).

GNNs generalize the CNN by replacing the standard convolutional filter with a so-called graph convolutional filter (Henaff et al., 2015; Gama et al., 2019; Ruiz et al., 2019). For a graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ with node and edge sets $\mathcal{V} = \{1, \ldots, N\}$, $\mathcal{E} = \{(i,j); i,j \in \mathcal{V}\}$ and weight function $\mathcal{W} : \mathcal{E} \to \mathbb{R}$, the graph shift operator (GSO) is defined as a matrix $S \in \mathbb{R}^{N \times N}$ that reflects the sparsity of the graph in the sense that $S_{ij} = 0$ if $i \neq j$ and $(i,j) \notin \mathcal{E}$. Common examples of GSOs are the adjacency matrix and the Laplacian matrix. Now let $y = [y^{(1)}, \ldots, y^{(N)}]$ a graph signal with components $y^{(i)}$ at node $i \in \mathcal{V}$ and $\psi \in \mathbb{R}^K$ a graph filter with coefficients $\psi = [\psi^{(1)}, \ldots, \psi^{(K)}]$. The application of filter ψ to graph signal y produces a vector $z \in \mathbb{R}^N$ with components

$$z_j := [\psi *_S y] = \sum_{k=0}^{K} \psi^{(K)} [S^K y]_j. \tag{16}$$

Note that the graph convolution in (16) reflects the local structure of the graph; we will have $S^K_{ij} \neq 0$ only if node j is a k-hop neighbor of i (Kipf and Welling, 2016).

GNNs are then composed of a series of (hidden) layers that combine graph convolutions and nonlinear operations. Each hidden layer l takes as input a graph signal y_l produced by the previous layer and outputs a graph signal y_{l+1} calculated by a

graph convolution followed by a nonlinear operation,

$$y_{l+1} = \phi_l \left(\psi_{l*S} y_l \right). \tag{17}$$

The nonlinear operation ϕ_l may be any function that respects the local structure in S (Ruiz et al., 2019).

Recall the random fading interference patterns of the wireless network underling the control system at time t as denoted by H_t. We may use the matrix H_t to define a graph $\mathcal{G}_t$ with nodes $\mathcal{V} = \{1, \dots, m\}$ given by the m plants and edges weighted by a function $\mathcal{W}_t((i,j)) := h_t^{(ij)}$ with GSO operator $S_t := H_t$—see Fig. 2. Observe that while in standard applications of GNNs, e.g. (Gama et al., 2019), the graph $\mathcal{G}$ is fixed, here the graph defined by the interference model in (3) is randomly distributed. Hence we make use of the notion of random edge GNNs (REGNNs) introduced by (Eisen and Ribeiro, 2019). Further define the input graph signal to be the plant states $y_0 = [x_t^{(1)}, \dots, x_t^{(m)}]$. The output of the REGNN is then given by

$$z_t = \phi_L \left(\psi_{L*H_t} \left(\dots \phi_1 \left(\psi_{1*H_t} x_t \right) \dots \right) \right). \tag{18}$$

This output may then be used to parameterize the policy distribution $\pi(p \mid s_t; \theta)$ in (11)—e.g. success probability of a Bernoulli distribution—where $\theta := [\psi_1, \dots, \psi_L]$ contains the filter coefficients that define the REGNN. Note that, similar to the case with CNNs, here we need to learn only the coefficients of the graph filters used at each hidden layer; letting K_l the filter length at each layer $l = 1, \dots, L$, the overall number of parameters we need to learn will be $n_p = \sum_{l=1}^{L} K_l$. Contrary to standard neural networks, n_p is independent of m.

The resulting procedure is shown in Algorithm 1.

Algorithm 1: RL/ GNNs for resource allocation in wireless control systems (adapted from (Sutton and Barto, 2018)).

Required: Horizon T; number of episodes N. System dynamics, communication model and cost function $J(x, u, \alpha)$ are used to simulate the environment but are unknown to the agent.

Result: Resource allocation policy.

initialization: load initial training / parameter set Θ

`/* loops over episodes */`

for $ii = 1, \dots, N$ **do**

  `/* T-step horizon simulation */`

  generates complete episode using current policy:

  $x_0, H_0, \alpha_0, r_1, \dots, x_{T-1}, H_{T-1}, \alpha_{T-1}, r_T$

  while $t < T$ **do**

    `/* calculates cost-to-go */`

    $R_t \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$

    `/* updates policy parameters */`

    $\theta \leftarrow \theta - \beta \gamma^t R_t \nabla \ln \pi(\alpha_t | s_t; \theta)$

  end

end

5. NUMERICAL EXPERIMENTS

We now present some numerical experiments to illustrate the use of reinforcement learning in combination with graph neural networks for resource allocation in wireless control systems. We consider a system made up by m scalar, independent, unstable plants sharing a total power budget $p_{\max}$. Plant dynamics

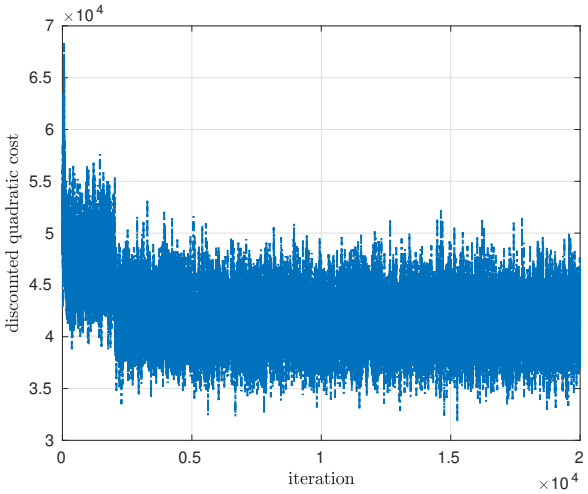

Fig. 3. Finite horizon cost (7) during learning phase.

are linear and randomly sampled from the interval $[1.01, 1.2]$ The channel states $h^{(ij)}$ consist of a path fading term $h_p^{(ij)}$ depending on the distance between transmitter and receiver, and a fast fading component $h_f^{(ij)}$ randomly sampled from a Rayleigh distribution with parameter $\mu = 1$. Thus, the fading and interference conditions experienced by plant i are given by

$$h^{(ij)} = h_p^{(ij)} h_f^{(ij)}. \tag{19}$$

For the numerical experiments we consider that the controller does not act when the transmission fails, that is, $\tilde{u}_t^{(i)} = 0$, and use a standard REINFORCE algorithm (Williams, 1992; Sutton et al., 2000) to learn an allocation policy. Moreover the resource allocation decision is based on estimates of plant states, that is the allocation policy receives as inputs $\tilde{x}_t$ with

$$\tilde{x}_t = x_t + w_t^{(o)} \tag{20}$$

where $w_t^{(o)}$ is an observation noise term with covariance matrix W_{obs}. Here, we took $W_{\text{obs}} = 1$ and $\sigma = 1$ (4). During the training phase we considered a finite horizon $T = 10$, but during the test phase we considered a longer simulation time $T = 30$. At each iteration of the algorithm we used a sample with batch size of 100 and adopted a learning rate $\beta = 5 \times 10^{-4}$.

First we considered a setting where each plant has a dedicated receiver, leading to an ad-hoc network. Plants are spatially distributed according to an uniform distribution $\mathcal{U}(-m, m)$, with receivers randomly assigned to positions sampled from $\mathcal{U}(-m/10, m/10)$ around the corresponding transmitter. The interference and fading conditions are computed according to (3), and the plants share a total power budget $p_{\max} = \frac{m}{2}$. Moreover, the initial states of the plants follow a normal distribution with mean 0 and variance 5, i.e. $x_0 \sim \mathcal{N}(0, 5)$. Figures 3 and 4 show the evolution of the cost at each iteration during the training phase for $m = 30$ plants. As expected, the overall cost of each episode decreases as the agent collects more experience, but the algorithm converged rather quickly when the policy is parameterized with (random edge) Graph Neural Networks. For this problem we considered a multivariate Gaussian policy with means scaled by a softmax procedure so as to respect the instantaneous power budget (7). After the training phase, the learned allocation policy was tested with a larger simulation horizon ($T = 30$) and compared with some baseline heuristics, namely dividing power equally among all the plants and ran-

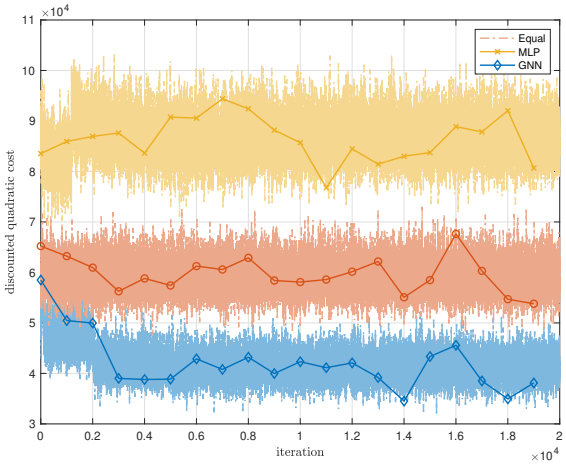

Fig. 4. Finite horizon cost (7) during learning phase for leaned policy using GNNs (blue), learned policy using multilayer perceptrons (yellow) and dividing power equally among all plants (orange).

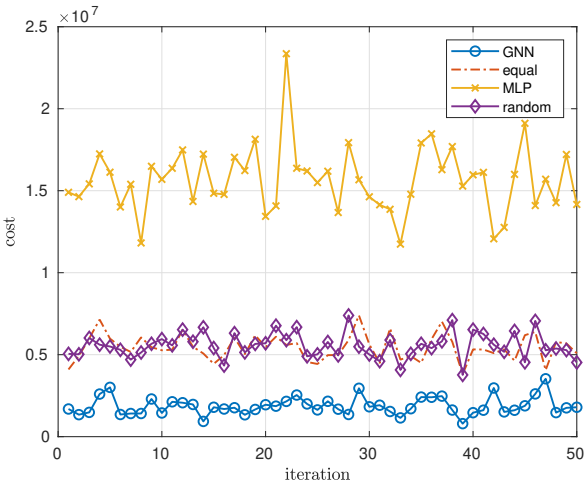

Fig. 5. Cost comparison between the learned allocation using GNNs (blue), learned allocation using multilayer perceptrons (yellow), distributing power equally among the plants (orange), and distributing power randomly according to a Dirichlet distribution (purple).

domly allocating power according to a Dirichlet distribution. The comparison is summarized in Figure 5. Notice that the learned allocation policy using GNNs performs better than all the heuristics in this setting, whereas the policy parameterized with fully connected neural networks or multilayer perceptrons (MLPs) does not perform well possibly due to the dimensionality of the inputs (m plant states and m^2 channel interference conditions) as well as the outputs (continuous allocation decision for m plants) in this setting.

Next we considered the multi-cell problem described in Section 2. In our simulations we considered first a system made up by $n = 5$ base stations with $k = 5$ users per station for a total of $m = 25$ plants. Base stations are distributed spatially according to an uniform distribution $\mathcal{U}(-m, m)$, and users are assigned to the nearest base. Note that, when compared with the ad-hoc setting, this problem is more challenging since the plants share

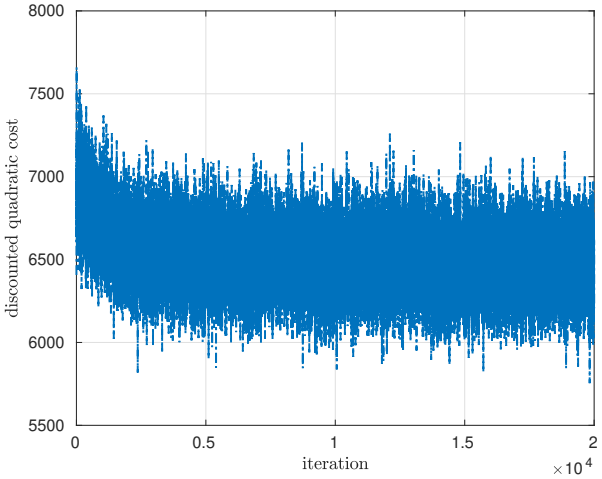

Fig. 6. Finite horizon cost during training (multi-cell configuration with 25 plants).

a transmitter and thus interference conditions make it difficult for nearby plants to receiver their packets without collision. In this setting, we consider then a type of scheduling problem, with the agent taking binary decisions and considering whether to send or not an information packet with some power p_0. The power allocation decision in this case takes the form

$$\alpha_t^i \in \{0, 1\} \tag{21}$$

with the corresponding SNIR given by

$$\text{SNIR}^{(i)} = \frac{h^{(ii)}\alpha^{(i)}(H)p_0}{\sigma^2 + \sum_{j \neq i} h^{(ji)}\alpha^{(j)}(H)p_0}. \tag{22}$$

To model this policy, we consider a Bernoulli distribution where the success probability for each plant is output by a graph neural network. Furthermore we took $p_0 = 7$, $\sigma = 1$ and $W_{\text{obs}} = 1$. Figure 6 shows the evolution of the finite horizon, quadratic cost (7) during training phase with a learning rate $\beta = 5 \times 10^{-4}$ and optimization horizon $T = 10$. As the number of iterations increases, the agent collects more experience and performance improves. REINFORCE, however, has rather poor sample complexity, and we expect to see faster convergence and improved performance when using state-of-the-art reinforcement learning algorithms.

After the learning phase, we compared the learned policy against some baseline heuristics and with a longer simulation horizon $T = 30$. Results are summarized in Figure 7, where we show the finite horizon quadratic cost (7) for the learned policy using GNNs (blue), learned policy using multilayer perceptrons (yellow), randomly selecting some plants to transmit (purple) and allowing plants to transmit when their state is above a certain threshold (orange). Each point in the graph corresponds to an average of 100 simulations with initial points x_0^i sampled from a normal distribution $\mathcal{N}(0, 5)$. In this setting, learned policies parameterized with GNNs and MLPs outperformed the baseline heuristics.

Next we scaled simulations to consider a setting with 15 base stations and 5 users per station. Simulation results are summarized in Figures 8 and 9. Figure 8 shows the finite horizon cost per iteration during the training phase for the policy parameterized with GNNs and MLPs. Using GNNs not only makes convergence faster, as can be seen in Figure 8, but also provides overall better performance when compared against

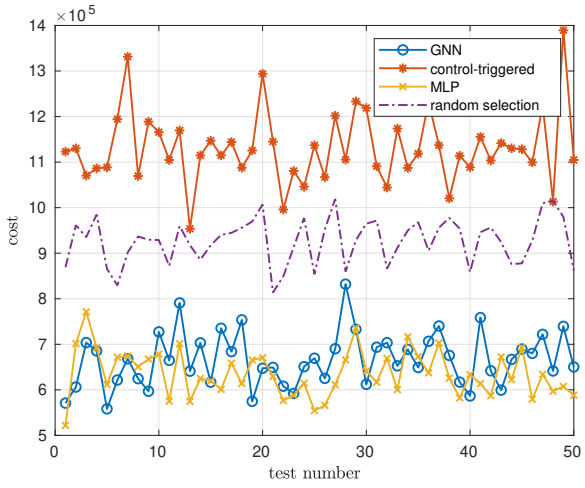

Fig. 7. Cost comparison between the learned allocation using GNNs (blue), learned allocation using standard neural networks / multilayer perceptrons (yellow), randomly choosing some plants to transmit (purple), and transmitting when the plant state is above a certain threshold (orange) for a multi-cell network with 5 base stations and 5 users per station.

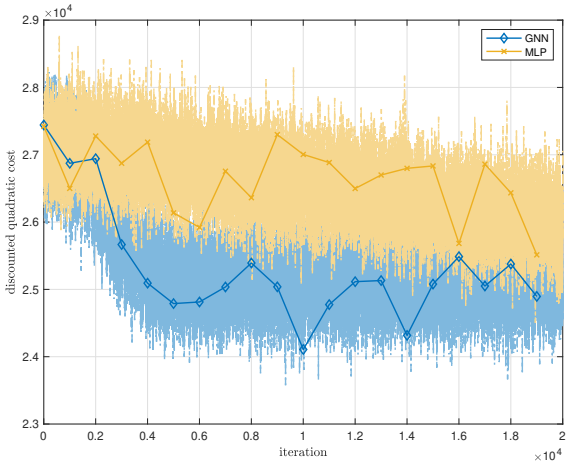

Fig. 8. Finite horizon cost during training for the policy parameterized with GNNs (blue) and standard neural networks (yellow) for a multi-cell configuration with 75 plants.

both baseline solutions and a policy parameterized with MLPs in this larger scale scenario as Figure 9 shows.

6. CONCLUSION

In this paper we discussed the use of reinforcement learning in tandem with graph neural networks (GNNs) to learn model-free allocation policies in wireless control systems. Resource allocation problems are usually challenging, leading to the recent widespread use of machine learning techniques to approximate optimal allocation policies. Deep reinforcement learning in particular has achieved significant success in this setting, but sampling complexity limits the applicability of traditional deep RL algorithms to large-scale wireless control problems. Here, we proposed to use GNNs instead of deep neural networks to parameterize the resource allocation policy. GNNs depend on

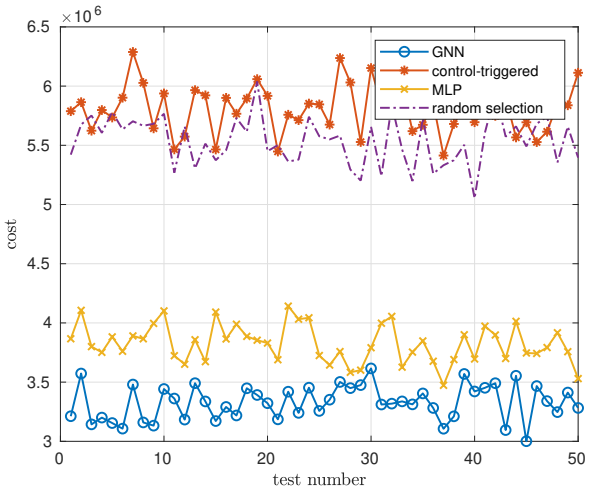

Fig. 9. Cost comparison between the learned allocation using GNNs (blue), learned allocation using multilayer perceptrons (yellow), randomly choosing some plants to transmit (purple), and transmitting when the plant state is above a certain threshold (orange) for a multi-cell network with 15 base stations and 5 users per station.

a smaller number of parameters while exploring local structure in the underlying communication network, allowing us to scale simulations to higher dimension settings. Numerical results show strong performance of the proposed approach when compared with baseline solutions. Moreover, the allocation policy parameterized with GNNs performed as good as or even better than a policy parameterized with standard neural networks, especially in very large scale settings. It is worth pointing out that the problems described here might be more adequate to a distributed setting, where local agents share some information structure but decide locally how to distribute resources among the plants in their vicinity. That is a problem that we plan to explore next, with graph neural networks making a natural candidate to learn resource allocation policies in that distributed setting.

REFERENCES

Dominik Baumann, Jia-Jie Zhu, Georg Martius, and Sebastian Trimpe. Deep reinforcement learning for event-triggered control. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 943–950, Dec 2018.

Christopher M. Bishop. *Chapter 5 - Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 2006.

Yaxin Cao and Victor O. K. Li. Scheduling algorithms in broadband wireless networks. *Proceedings of the IEEE*, 89(1):76–87, Jan 2001.

Themistoklis Charalambous, Ayca Ozcelikkale, Mario Zanon, Paolo Falcone, and Henk Wymeersch. On the resource allocation problem in wireless networked control systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4147–4154, December 2017.

Burak Demirel, Arunselvan Ramaswamy, Daniel E. Quevedo, and Holger Karl. DeepCAS: A Deep Reinforcement Learning Algorithm for Control-Aware Scheduling. *IEEE Control Systems Letters*, 2(4):737–742, October 2018.

Mark Eisen and Alejandro Ribeiro. Optimal wireless resource allocation with random edge graph neural networks, 2019.

Mark Eisen, Clark Zhang, Luiz F. O. Chamon, Daniel D. Lee, and Alejandro Ribeiro. Learning optimal resource allocations in wireless systems. *IEEE Transactions on Signal Processing*, 67(10):2775–2790, May 2019.

Atilla Eryilmaz and Rayadurgam Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *IEEE/ACM Transactions on Networking*, 15(6):1333–1344, Dec 2007.

Fernando Gama, Antonio G. Marques, Geert Leus, and Alejandro Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67(4):1034–1049, Feb 2019.

Konstantinos Gatsis, Miroslav Pajic, Alejandro Ribeiro, and George. J. Pappas. Opportunistic Control Over Shared Wireless Channels. *IEEE Transactions on Automatic Control*, 60(12):3140–3155, December 2015.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data, 2015.

Onesimo Hernandez-Lerma and Jean B. Lasserre. *Discrete-Time Markov Control Processes: Basic Optimality Criteria*. Stochastic Modelling and Applied Probability. Springer-Verlag, New York, 1996.

Joao P. Hespanha, Payam Naghshtabrizi, and Yonggang Xu. A Survey of Recent Results in Networked Control Systems. *Proceedings of the IEEE*, 95(1):138–162, January 2007.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.

Mengyuan Lee, Guanding Yu, and Geoffrey Ye Li. Graph embedding based wireless link scheduling with few training samples. *arXiv preprint arXiv:1906.02871*, 2019.

Alex S. Leong, Arunselvan Ramaswamy, Daniel E. Quevedo, Holger Karl, and Ling Shi. Deep Reinforcement Learning for Wireless Sensor Scheduling in Cyber-Physical Systems. *ArXiv e-prints*, September 2018.

Fei Liang, Cong Shen, Wei Yu, and Feng Wu. Towards Optimal Power Control via Ensembling Deep Neural Networks. *arXiv e-prints*, art. arXiv:1807.10025, Jul 2018.

Vinicius Lima, Mark Eisen, Konstantinos Gatsis, and Alejandro Ribeiro. Resource allocation in wireless control systems via deep policy gradient, 2020. To be presented at the 2020 IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC).

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

Pangun Park, Sinem Coleri Ergen, Carlo Fischione, Chenyang Lu, and Karl Henrik Johansson. Wireless Network Design for Control Systems: A Survey. *IEEE Communications Surveys Tutorials*, 20(2):978–1013, 2018.

Henrik Rehbinder and Martin Sanfridson. Scheduling of a limited communication channel for optimal control. *Automatica*, 40(3):491–500, March 2004.

Luana Ruiz, Fernando Gama, Antonio G. Marques, and Alejandro Ribeiro. Invariance-preserving localized activation functions for graph neural networks, 2019.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, Jan 2009.

Yifei Shen, Yuanming Shi, Jun Zhang, and Khaled B Letaief. A graph neural network approach for scalable wireless power control. *arXiv preprint arXiv:1907.08487*, 2019.

Ling Shi, Peng Cheng, and Jiming Chen. Optimal Periodic Sensor Scheduling With Limited Resources. *IEEE Transactions on Automatic Control*, 56(9): 2190–2195, September 2011.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. The MIT Press, second edition edition, November 2018.

Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *In Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.

Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, jan 2010.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.