Multi-agent Reinforcement Learning with Graph Q-Networks for Antenna Tuning Appendix

Maxime Bouton, Jaeseong Jeong, Jose Outes, Adriano Mendo, Alexandros Nikou

APPENDIX

The appendix contains additional details about the method, parameters used for the experiments and additional experiments.

A. EXPERIMENTAL DETAILS

In this appendix, we describe the different hyperparameters for the algorithms and some implementation details about the experiments. Our simulator models an LTE network with antennas that can be controlled remotely for changing the electrical downtilt angle and the maximum downlink transmitted power and operating at a frequency of 2 GHz. The network has 10 000 users uniformly distributed on the map, generating an average traffic of 1 Mbps per cell.

The training consists of 20 000 steps split into episodes of 20 steps. At the beginning of an episode, a new deployment is sampled. We consider 19 base stations in the training environment, each consisting of 3 antennas, which makes a total of 57 cells. Each cell is associated to a learning agent. The average intersite distance between base stations is uniformly sampled between 300 m and 1500 m. We consider both hexagonal deployment and randomly generated deployment of the base stations with a minimum intersite distance, as illustrated in Fig. 1. At the beginning of an episode, the electrical tilt of each antenna is reset to a random value within the range $[0°, 15°]$. In the joint tilt and power control experiment the power is also reset to a random value while for the tilt experiment it is fixed at 40 W.

The baselines are using exactly the same simulator configuration and are trained with the same random seed such that they experience exactly the same simulated networks both in training and evaluation. The observation features and the reward signal are normalized during training. They all use an ϵ-greedy policy for exploration with a decay of the exploration rate, ϵ, from 1 to 0.01 during the first half of the training. Each training is repeated with three different random seeds.

All the baselines are implemented using rllib [1] and the Pytorch geometric library [2], they all use a target network, prioritized replay, double Q-learning and distributed experience collection with 5 workers, each using 3 CPUs. We report all the hyperparameters in Table I. For all the baselines we carried out a hyperparameter search on the learning rate and number of layers and we rescaled the observation vectors and reward signals such that their value is between $[-1, 1]$. We used a discount factor of 0.0 since our problem's goal, reaching the final optimal antenna configuration, does not require visiting intermediate states with performance degradation. Increasing the discount factor to 0.9 did not lead to any benefits for any of the baselines for this specific application. However, the

The authors are with Ericsson. Email: {maxime.bouton, jaeseong.jeong, jose.outes, adriano.mendo, alexandros.nikou}@ericsson.com

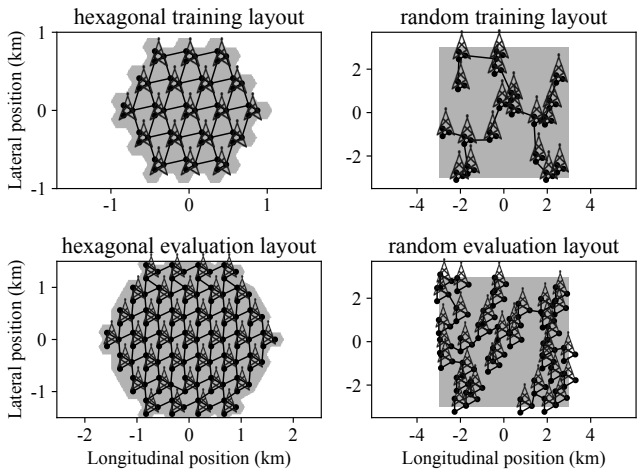

Fig. 1. Visualization of the hexagonal and random deployments used for training and evaluation. Macro base station network deployments used in our experiments with the graph connecting the different cells (three per base station). The topology of the graph can change depending on the intersite distance. Evaluation layouts use denser deployments with more agents. The position of the node is offset from their real position for better visualization.

TABLE I
HYPERPARAMETERS FOR TRAINING THE BASELINES AND OUR ALGORITHM. THE FIRST LINES ARE COMMON TO ALL ALGORITHMS.

γ	0.0
learning rate	1×10^{-3}
initial ϵ	1.0
final ϵ	0.01
ϵ decrease steps	10 000
activation function	ReLU
batch size	64
(N)DQN architecture	FC(64), FC(32)
GAQ architecture	GAT(32, 6 heads), GAT(32, 6 heads), FC(32), FC(32)
GQN architecture	GCN(32), GCN(32), FC(32), FC(32)
GQN(GAT) architecture	GAT(32, 4 heads), GAT(32, 4 heads), FC(32), FC(32)
GQN learning rate	1×10^{-2}

method, as described in the previous section, still holds for larger discount factors. It leads to the same asymptotic results but with slower convergence.

In the N-DQN baseline, a maximum of 5 neighbors is considered. Each observation vector from the neighbors is stacked and fed to a feed forward neural network.

For our GQN algorithm, we tried adding encoding MLPs before the GNN layer but it did not bring any improvement in performance (nor did it damage it). We also experimented with a graph convolutional neural network layer instead of the graph attention one.

We provide a pseudocode of our proposed GQN method in Algorithm 1. It follows a similar off policy training procedure as DQN. We use a target network, double Q learning and prioritized experience replay. The state of each agent and the graphs are stored in the replay buffer.

Algorithm 1 GQN training procedure.

Initialize: GQN weights, replay buffer $\mathcal{D} = \{\}$, training steps T, batch size B, ϵ schedule
for $t = 1, \dots T$
  Observe joint state $\mathbf{s}$, and adjacency matrix A
  With probability ϵ, choose a random joint action $\mathbf{a}$
  Otherwise choose $\mathbf{a} = \mathbf{a}^*$ according to **??**.
  Observe the next joint state $\mathbf{s}$, the next adjacency matrix A', and the global reward r.
  Store the transition $(\mathbf{s}, A, \mathbf{a}, r, \mathbf{s}', A')$ in $\mathcal{D}$.
  Sample batch $\{(\mathbf{s}^k, A^k, \mathbf{a}^k, r^k, \mathbf{s'}^k, A'^k)$ for $k = 1, \dots, B\}$ from $\mathcal{D}$.
  Assign $y^k = r^k + \gamma \max_{\mathbf{a}'} \sum_i Q_i(\mathbf{s'}^k, A'^k, a'_i) \forall k$ as per **??**.
  Perform a gradient descent step on the loss $1/B \cdot \sum_k [(\sum_i Q_i(\mathbf{s}^k, A^k, a_i^k) - y^k)^2]$
end for

B. Additional Experiments

Tilt Generalization

In the tilt generalization experiment, we tried different combination of training the models on random deployments and evaluating on hexagonal deployments and vice versa. The results are presented in Fig. 2. In all cases, the GQN-GAT method present the best performance. An interesting outcome is that the GQN(GAT) model trained on random deployment, has a better performance on the hexagonal evaluation scenario (bottom left plot) than the DQN and GAQ models that were trained specifically on hexagonal topologies (top left plot). On the evaluation with random cell layout, the heuristic has a very close performance to some of the reinforcement learning algorithms, indicating that generalization to those scenarios is a difficult task, especially when the difference between training and deployment layout is large (top right plot).

Joint Tilt and Power Control

In this scenario the reward function is parameterized by w which controls the trade-off between minimizing power and maximizing signal quality. We experimented with value of w in $\{0.05, 0.1, 0.15, 0.2, 0.5\}$ for the reward functions. For the most extreme values for w the agents learns to set the power to the maximum value (low w), or the minimum (high w). The values of 0.15 and 0.2 gave the most interesting results where the agent converged to non extreme power values. We witnessed the same effect for all methods and only report the results for $w = 0.15$ for the sake of readability.

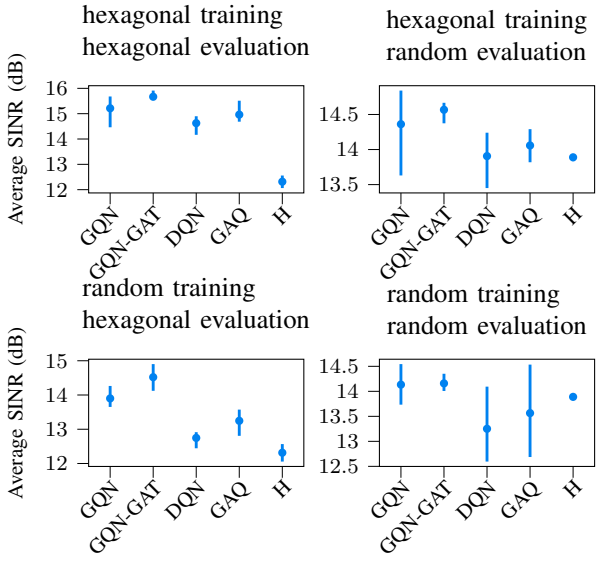

Fig. 2. Generalization performance of the GQN algorithm and the baselines on deployments unseen at training time. The evaluation deployments are denser, as illustrated in **??**. The points represent the average performance and the error bars represent the 95th percentile confidence interval (sometimes smaller than the marker).

In Fig. 3, we add the evolution of the global reward. The GQN model are directly trying to optimize this reward while other algorithms must rely on local signals. As a consequence they do not give as much power reduction as GQN.

C. RSRP and SINR Calculations

To derive the relation between SINR and cell configurations we first define the reference signal received power (RSRP) $R_{c,u}$ for a user u connected to a cell c:

$$R_{c_u}(\phi, \alpha) = \frac{P_c(\phi) G_{c,u}(\alpha)}{L_{c,u}} \tag{1}$$

where $P_c(\phi)$ is the transmitted power of the antenna per reference signal resource element, as a function of the maximum transmitted power ϕ. $G_{c,u}(\alpha)$ is the antenna gain which depends on the tilt angle α and the azimuth which is kept fixed in our model. $L_{c,u}$ is the path loss. To compute the antenna gains, one can use the relationship defined in the third generation partnership project [3]. The maximum power ϕ is divided by the number of available resource blocks and multiplied by the cell specific reference signal power boost gain (set to one in our simulation). The received power is only depending on the cell that the user is connected to and on the propagation environment which affects the path loss. The

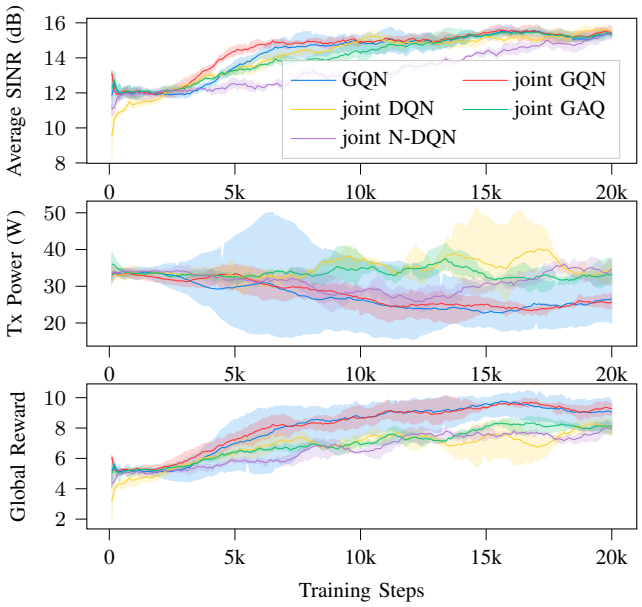

Fig. 3. Training performance of GQN (including global reward) with separate agents for tilt and power, joint GQN and the baselines in the joint control environment. For a similar average SINR, GQN is able to yield a greater power reduction. The solid lines represent the mean over 3 random seeds and the shaded area is the 95th percentile confidence interval.

user also receives power from other cells which is regarded as interference in the SINR calculation. For a user u, the cell c that the user is attached to is assumed to be the cell yielding the largest received power. The downlink SINR of a user u is defined as the ratio between the received power from cell c and the sum of the received power from all other cells, and the noise power μ:

$$\rho_u(\phi, \alpha) = \frac{R_{c_u}(\phi_c, \alpha_c)}{\sum_{i=1, i\neq c}^{N_{\text{cells}}} R_{i,u}(\phi_i, \alpha_i) + \mu} \tag{2}$$

The noise power is calculated over a frequency bandwidth of one resource element (15 kHz). Improving the SINR of a user involves improving the received power as well as reducing interference from other cells.

References

[1] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. I. Jordan, and I. Stoica, "Rllib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2018.

[2] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[3] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Further advancements for E-UTRA physical layer aspects," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.814, 2017, Version 9.2.0.