Deep Reinforcement Learning Based Resource Allocation in Wireless Networks

Yasar Sinan Nasir
Advised by Prof. Dongning Guo

Department of Electrical and Computer Engineering
Northwestern University, Evanston, IL 60208.

October 21, 2021

Outline

1. Introduction
2. Radio Resource Management Problem
3. Multi-Agent Deep Reinforcement Learning (DRL) based Solution
4. Full-buffer Simulations
5. Traffic Simulations
6. Conclusion and future work

AI for Wireless:

- Data-driven, model-free AI for wireless can learn sophisticated strategies to enhance the network performance by processing limited previous data.
- The industry is very interested in AI based wireless resource management.

Progress:

- Power Control:
  Multi-Agent Deep Reinforcement Learning for Dynamic Power Allocation in Wireless Networks (IEEE JSAC 2019)
  - Adding Mobility:
    Deep Actor-Critic Learning for Distributed Power Control in Wireless Mobile Networks (Asilomar 2020)
- Joint Spectrum and Power Allocation:
  Deep Reinforcement Learning for Joint Spectrum and Power Allocation in Cellular Networks (submitted to Globecom 2021)
- Both Varying Traffic and Channel Conditions
  Traffic-driven Radio Resource Management via Deep Reinforcement Learning (to be submitted)

System Model:

- N links, K cells, SISO, M subbands.
- $\mathcal{K} = \{1, \ldots, K\}$, $\mathcal{N} = \{1, \ldots, N\}$, and $\mathcal{M} = \{1, \ldots, M\}$
- If link n's user is inside cell k, its associated base station $b_n \in \mathcal{K}$ is at the center of cell k.
- Base station b_n transmits to user n over m in time slot t with $p_{n,m}^{(t)} \geq 0$.
- The power constraint restricts the total transmit power on subband m:

$$\sum_{n \in \mathcal{N} : b_n = k} p_{n,m}^{(t)} \leq P_{\max}, \forall k, \forall m.$$

Link Model:

- $g^{(t)}_{b_n \rightarrow n,m}$: direct channel gain.
- $g^{(t)}_{b_j \rightarrow n,m}$: interfering channel gain.
- Spectral efficiency:

$$C^{(t)}_{n,m}\left(\boldsymbol{p}^{(t)}_m\right) = \log\left(1 + \frac{g^{(t)}_{b_n \rightarrow n,m} p^{(t)}_{n,m}}{\sum_{j \in \mathcal{N}, j \neq n} g^{(t)}_{b_j \rightarrow n,m} p^{(t)}_{j,m} + \sigma^2}\right),$$

where $\boldsymbol{p}^{(t)}_m = \left[p^{(t)}_{1,m}, p^{(t)}_{2,m}, \ldots, p^{(t)}_{N,m}\right]^{\mathsf{T}}$.

-

$$C^{(t)}_n = \sum_{m \in \mathcal{M}} C^{(t)}_{n,m}\left(\boldsymbol{p}^{(t)}_m\right).$$

Traffic Model:

- Each link has a queue.
- $N_n^{(t)}$ is link n's queue length in bits at the beginning of time slot t.
- W : total bandwidth.
- T : time slot duration.
- $A_n^{(t)}$: newly arrived packets.
- $$N_n^{(t)} = \max\left(N_n^{(t-1)} - C_n^{(t-1)}WT, 0\right) + A_n^{(t)}L,$$

Channel Variations:

The downlink channel gain:

$$g^{(t)}_{b_n\to n,m} = \beta_{b_n\to n}\left|R^{(t)}_{b_n\to n,m}\right|^2, \quad t = 1, 2, \dots ,$$

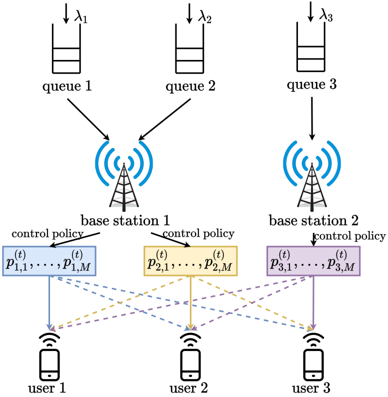
where

- $\beta_{b_n\to n} \geq 0$ is the large-scale fading that includes path loss and log-normal shadowing. It is same across all subbands.
- $R^{(t)}_{b_n\to n,m}$ is the small-scale fading. It is is frequency selective and modeled by Jakes' Model:

$$R^{(t)}_{b_n\to n,m} = \rho R^{(t-1)}_{b_n\to n,m} + \sqrt{1-\rho^2}e^{(t)}_{b_n\to n,m},$$

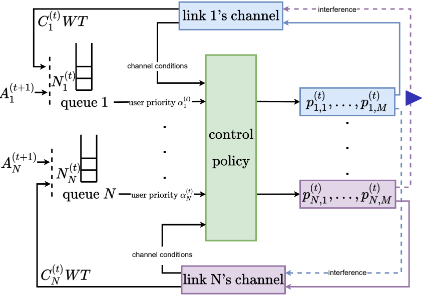
where $\rho = J_0(2\pi f_d T)$, $R^{(0)}_{b_n\to n,m} \sim \mathcal{CN}(0,1)$, and $e^{(1)}_{b_n\to n,m}, e^{(2)}_{b_n\to n,m}, \dots$ consists of i.i.d. CSCG random variables with unit variance.

The Fundamental Problem:

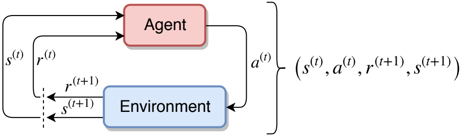


- A control policy that maps traffic & channel conditions to physical layer allocations.
- The long-term utility of user n, U_n, should reflect the average packet delay.
- The fundamental problem becomes finding an optimal control policy that maximizes $U = \sum_{n \in \mathcal{N}} U_n$.

The Fundamental Problem and Reinforcement Learning



- Model-free reinforcement learning learns directly from trial-and-error-interactions:



- The policy $\pi(a|s)$ denotes the probability of taking action a conditioned on the current state being s. The Q-function:

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[R^{(t)} \middle| s^{(t)} = s, a^{(t)} = a \right]$$

where $R^{(t)} = \sum_{\tau=0}^{\infty} \gamma^\tau r^{(t+\tau+1)}$ and $\gamma \in (0,1]$.
- For radio resource management, $r^{(t+1)}$ can be thought as $-\sum_{n \in \mathcal{N}} N_n^{(t+1)}$.

Deep Q-learning algorithm

- indirectly optimize agent performance by learning a value function.
- Use a deep Q-network (DQN) parameterized by $\boldsymbol{\psi}$ to represent the Q-function values $q(\cdot, \cdot; \boldsymbol{\psi})$
- It is an off-policy learning that stores experiences in a memory $\mathcal{D}$.
- $\boldsymbol{\psi}$ is updated using a stochastic gradient descent algorithm by

$$\nabla_{\boldsymbol{\psi}} \frac{1}{|\mathcal{B}|} \sum_{(s,a,r',s') \in \mathcal{B}} \left(y(r', s') - q\left(s, a; \boldsymbol{\psi}\right)\right)^2,$$

where the target is $y(r', s') = r' + \gamma \max_{a'} q\left(s', a'; \boldsymbol{\psi}_{\text{target}}\right).$

Conventional Divide-and-Conquer Solution

- Non-negative user weights (priorities), $\alpha_n^{(t)}, \forall n \in \mathcal{N}$, are used to separate the network layer problem and the physical layer problem:

$$\begin{aligned} \underset{\boldsymbol{p}^{(t)}}{\text{maximize}} \quad & \sum_{n=1}^{N} \alpha_n^{(t)} \sum_{m \in \mathcal{M}} C_{n,m}^{(t)} \left(\boldsymbol{p}_m^{(t)}\right) \\ \text{subject to} \quad & p_{n,m}^{(t)} \geq 0, \forall n \in \mathcal{N}, \, m \in \mathcal{M}, \\ & \sum_{j \in \mathcal{N}_k} p_{j,m}^{(t)} \leq P_{\max}, \forall\, k \in \mathcal{K}, \, m \in \mathcal{M}. \end{aligned}$$

- Proportionally fair scheduling: Recall [Tse and Viswanath '05]:

$$\alpha_n^{(t+1)} = 1/\bar{C}_n^{(t)},$$

where $\bar{C}_n^{(t)} = \beta \cdot C_n^{(t)} + (1-\beta)\bar{C}_n^{(t-1)}$ with $\beta \in (0, 1]$.
This scheme maximizes:

$$\sum_{n \in \mathcal{N}} \log \bar{C}_n^{(t)}.$$

Existing Solutions:

- Conventional Optimization Based:
  - Weighted MMSE (WMMSE) [Shi, Razaviyayn, Luo, and He '11]
  - Fractional programming (FP) [Shen and Yu '18]
  - State of the art when
    - a mathematically tractable accurate system model is available;
    - full channel state information (CSI) is available;
    - iterations converge instantly (for time-varying channels);
    - no network backhaul latency (for time-varying channels).
- Deep Learning Based Solution: [Sun, Chen, Shi, Hong, Fu, and Sidiropoulos '17] proposed a centralized supervised learning scheme;
  - Trains a faster deep neural network (DNN) to approximate WMMSE;
  - Achieves 90% or higher of the sum-rate achieved by WMMSE.

Literature on reinforcement learning for power control:

- [Bennis and Niyato '10] and [Simsek, Czylwik, Galindo-Serrano, and Giupponi '11] used classical Q-learning to reduce the interference in LTE-Femtocells.
- [Amiri, Mehrpouyan, Fridman, Mallik, Nallanathan, and Matolak '18] have used cooperative Q-learning to increase QoS of users in femtocells without considering channel variations.
- [Xu, Wang, Tang, Wang, and Gursoy '17] proposed a centralized deep reinforcement learning approach.
- [Calabrese, Wang, Ghadimi, Peters, and Soldati '17] proposed a similar distributively executed framework to us by using deep Q-learning. No channel variations.
- [Liang, Ye, and Li '19] applied deep Q-learning to minimize V2V links' interference to V2I links, channel variations are simulated by Jakes fading model similar to us.

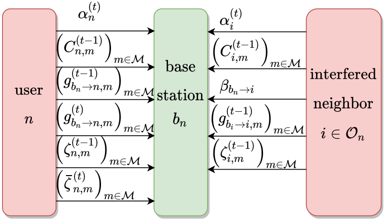
Our main contributions:

- Time varying traffic and channel conditions;
- Practicality constraints on measurements;
- Assume information exchange only between nearby links (delays);
- Distributively executed resource allocation;
- Flexible Objective: The agents collaboratively maximize a quality of service (QoS) objective over their local environment, that can be
  - the average packet delay, or
  - the sum rate, or
  - a proportionally fair throughput, or
  - anything else specified by the network layer.

Distributed Execution

- Centralized vs Distributed Execution:
  - × A centralized single learning agent that outputs joint actions by observing the complete environment state.
  - ✓ Multiple learning agents that output their own action by observing local environment.
- The environment transition is no longer stationary as other agents in the system update their policies/behaviors simultaneously. Multi-agent learning schemes have good empirical performance, but no theoretical guarantee.
- A global DQN is trained by the experiences of all agents.
- Training is centralized to ease implementation and to improve stability.

Preliminaries: Local Information & Aggregates



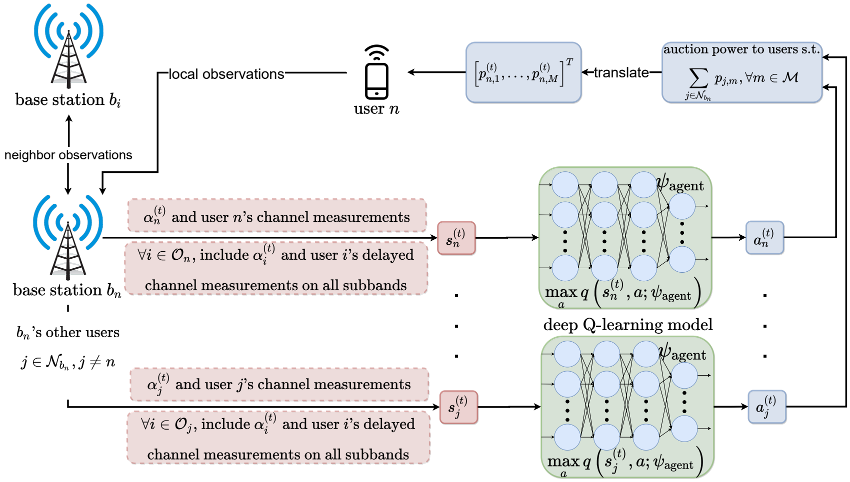
- Neighborhood set of n, $\mathcal{O}_n$, is the set of c receivers with largest $\beta_{b_n\to i}$.
- The aggregated interference at user n on subband m in time slot $t-1$:

$$\zeta_{n,m}^{(t-1)} = \sum_{j\in\mathcal{N},j\neq n} g_{b_j\to n,m}^{(t-1)} p_{j,m}^{(t-1)} + \sigma^2.$$

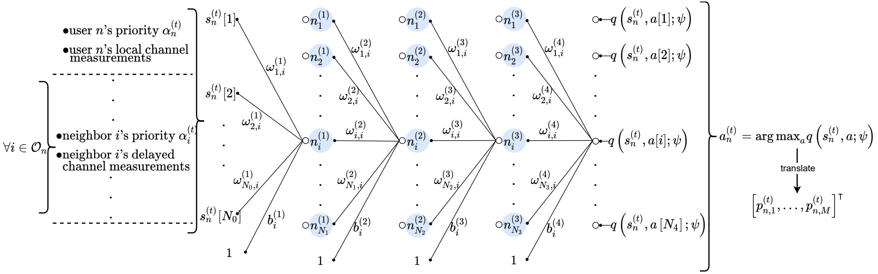
- The aggregated interference at the end of time slot $t-1$ with updated channel gains but with power allocation during $t-1$:

$$\bar{\zeta}_{n,m}^{(t)} = \sum_{j\in\mathcal{N},j\neq n} g_{b_j\to n,m}^{(t)} p_{j,m}^{(t-1)} + \sigma^2.$$

Proposed Distributed Execution Framework



Local State Set Design and The Policy



- Local state of agent n, $s_n^{(t)}$, is composed of:
  1. priority of user n, $\alpha_n^{(t)}$;
  2. most-recent channel measurements of user n;
  3. priorities and delayed channel measurements of all neighbors $\in \mathcal{O}_n$.

Local State / priority of user n

- Traffic-Aware Scheduling: $\alpha_n^{(t)}$ has two entries:
  1. total number of packets waiting in link n's queue;
  2. the rate estimate of link n for time slot t,

$$\bar{\lambda}_n^{(t)} = \frac{\sum_{\tau} \xi^{\tau} A_n^{(t-\tau)}}{\sum_{\tau=1}^{T_{\mathrm{r}}} \xi^{\tau}},$$

- Proportionally fair scheduling: $\alpha_n^{(t)} = 1/\bar{C}_n^{(t-1)}$.

Local State / most-recent channel measurements of user n

- M feature subgroups corresponding to M subbands.
- For subband m, reserve 6 entries:
  - $C_{n,m}^{(t-1)}$;
  - $p_{n,m}^{(t-1)}$;
  - last two measurements of the direct channel gains, $g_{b_n \rightarrow n,m}^{(t)}$ and $g_{b_n \rightarrow n,m}^{(t-1)}$;
  - last two aggregated interference measurements, $\bar{\zeta}_{n,m}^{(t)}$ and $\zeta_{n,m}^{(t-1)}$.

Local State / priorities and delayed CSI of neighbors

- For each neighbor $i \in \mathcal{O}_n$:
- neighbor i's priority: $\alpha_i^{(t)}$;
- neighbor i's significance $\beta_{b_n \to i}$;
- for subband m, 3 entries for neighbor i's delayed channel measurements:
  1. $C_{i,m}^{(t-1)}$;
  2. link i's direct channel gain, $g_{b_i \to i,m}^{(t-1)}$;
  3. most-recent aggregated interference measurement of user i that is available at base station b_n, $\zeta_{i,m}^{(t-1)}$.

Action Set

- Allowed actions on subband m:

$$\mathcal{A}_m = \left\{ 0, P_{\min}, P_{\min} \left(\frac{P_{\max}}{P_{\min}} \right)^{\frac{1}{|\mathcal{A}_m|-2}}, \ldots, P_{\max} \right\},$$

where $P_{\min}$ is the minimum positive transmit power level.

- The action space of agent n:

$$\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_M.$$

Reward function / Local Objective

- We enable collaboration by including signal from neighbors to agent's reward,

$$r^{(t+1)}_{\text{local objective,n}} = \pi_n^{(t)} + \sum_{i \in \mathcal{O}_n} \pi_i^{(t)},$$

where $\pi_n^{(t)}$ is agent n's direct contribution.

- For the traffic-aware scheduling, let:

$$\pi_n^{(t)} = -\max\left(N_n^{(t)} - C_n^{(t)} WT, 0\right).$$

- Alternatively, to maximize weighted sum-rate, let:

$$\pi_n^{(t)} = \alpha_n^{(t)} C_n^{(t)}.$$

Reward function / Externalities

- Ideally, we would use a reward function with externalities

$$r^{(t+1)}_{\text{externalities,n}} = \pi^{(t)}_n - \sum_{i \in \mathcal{O}_n} \pi^{(t)}_{n \to i},$$

where $\pi^{(t)}_{n \to i}$ is the externality from link n to neighbor i.
- The externality computation would require individual interfering channel gains from base station b_n to neighbor i, i.e., $g^{(t)}_{b_n \to i}$, $\forall i \in \mathcal{O}_n$.
- For example, for weighted sum-rate maximization,in time slot t

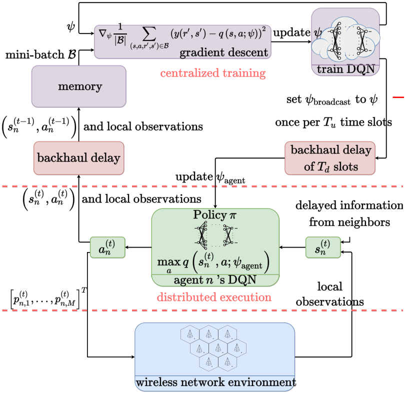
$$\pi^{(t)}_{n \to i} = \alpha^{(t)}_i \left(C^{(t)}_{i \backslash n} - C^{(t)}_i \right),$$

where $C^{(t)}_{i \backslash n}$ is the spectral efficiency of neighbor i without the interference from link n: $C^{(t)}_{i \backslash n} = \sum_{m=1}^{M} \log \left(1 + \frac{g^{(t)}_{b_i \to i,m} p^{(t)}_{i,m}}{\zeta^{(t)}_{i,m} - g^{(t)}_{b_n \to i,m} p^{(t)}_{n,m}} \right)$.

Episodic Training Scheme with Varying Traffic Load

- Goal is to train a single policy to handle various traffic load conditions in the execution stage without further adjustment on the policy.
- The proposed episodic training scheme is composed of multiple consecutive episodes with each episode having a random wireless network initialization and an average arrival rate (traffic load) λ_{avg}.
- Inside each episode, training is structured as a series of interactions between two algorithms namely "Distributed exection" and "Centralized Training". These interactions occur on a time scale of 1 time slot.
- Training samples a mini-batch from global memory $\mathcal{D}_{\text{g}}$ and experience-replay memory $\mathcal{D}$ of current episode.
- At the end of each time slot, training checks for the queue stability.
- If queues remain stable for $T_{\max}$ time slots, training moves to next episode. If resulting average delay is converged, λ_{avg} is increased by λ_{inc}.

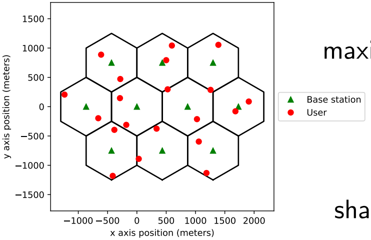
Centralized training and distributed execution framework:



1: **Centralized training** $(\psi, \psi_{\textbf{broadcast}}, \mathcal{D}_{\textbf{g}}, \mathcal{D})$**:**
2: Sample $\mathcal{B}$ from the experiences in $\mathcal{D}_{\text{g}}$ and $\mathcal{D}$.
3: Update ψ using a gradient descent step.
4: If it has been T_u since last policy broadcast, update $\psi_{\text{broadcast}}$ by ψ to update ψ_{agent}.
Output: Updated ψ, $\psi_{\text{broadcast}}$, ψ_{agent}.

1: **Parameters:** ϵ-greedy algorithm's ϵ.
2: **Distributed exec.** $(\psi_{\textbf{agent}})$ at time slot t:
3: **for** agent $n = 1, 2, \dots, N$ **do**
4: Agent n observes its current local state $s_n^{(t)}$;
5: sets $a_n^{(t)} = \arg\max_a q(s_n^{(t)}, a; \psi_{\text{agent}})$.
6: If $t \mod N = n - 1$, set $a_n^{(t)}$ to a random action with a probability of ϵ.
7: Translate action to $[p_{n,1}^{(t)}, \dots, p_{n,M}^{(t)}]^\intercal$, after auction at base station b_n.
8: **end for**
Output: $p_m^{(t)} \forall m \in \mathcal{M}$ & $\left(s_n^{(t)}, a_n^{(t)}\right) \forall n \in \mathcal{N}$.

Simulation Setup



maximum transmit power $P_{\max}$	38 dBm
total bandwidth	10 MHz
slot duration T	20 ms
traffic pattern	full-buffer
path loss (in dB)	$120.9 + 37.6 \log_{10}(d)$
shadowing standard deviation	8 dB
AWGN power	-114 dBm

- DQN:
  - 3 hidden layers of 200, 100, and 50 neurons, respectively;
  - Fully connected; the activation function is $\tanh(\,)$;
  - Limited to 5 neighbors;
  - Limited to 10 discrete power levels.

DQN training

- The trainer broadcasts the new parameters once every 100 slots; these parameters are available at the agents after 50 slots; minimum required downlink/uplink capacity for all backhaul links is about 1 Mbps.
- $\mathcal{D}$ stores 1,000 most recent experiences from each link;
- Use RMSProp to train with a random mini-batch of 256 experiences.
- The proposed algorithms:
  1. Matched DQN – train and test on same deployment.
  2. Unmatched DQN – trained for a different network (different device locations and fading)
- Benchmark allocations:
  1. WMMSE (genie-aided with full instantaneous CSI)
  2. FP (genie-aided with full instantaneous CSI)
  3. centralized (FP with delayed full CSI)
  4. full-power (or max-power) allocation.

Sum-rate maximization: scalability

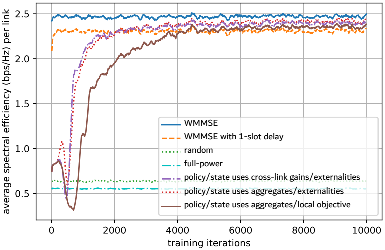
1 user per cell, M=1 subband, $R = 500$ m, $f_d = 10$ Hz.
(cross-link CSI is available to the DQN.)

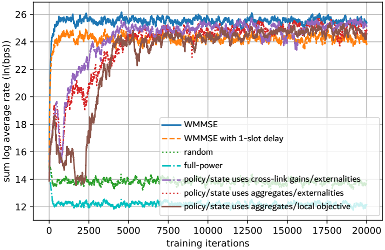
	average sum-rate in bps/Hz per link					
	DQN		benchmark power allocations			
N (links)	matched	unmatched	WMMSE	FP	central	full-power
19	2.78	2.50	2.66	2.58	2.44	1.37
50	2.28	1.99	2.17	2.13	2.00	1.02
100	1.92	1.68	1.90	1.88	1.74	0.89

- Each link determines its action within 0.3 ms.
- A single batch takes up to 17 ms (without GPU).
- FP requires about 15 ms to converge for $n = 19$ links, but with $n = 100$ links this becomes 35 ms.
- WMMSE converges slightly slower than the FP algorithm.

cross-link CSI vs aggregates

N = 20 links, K = 10 cells, M=1 subband, R = 500 m, f_d = 10 Hz.



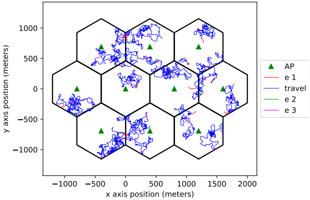


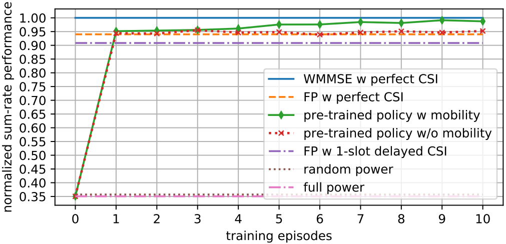
(left) sum-rate maximization; (right) proportionally fair scheduling.

	Average sum-rate performance in bps/Hz per link.				without aggregates.	
(cells,links)	DQN trained for (10,20)	WMMSE	FP	FP w delay	random	full
(10,20)	2.59; 99.2% of WMMSE	2.61	2.45	2.37	0.93	0.91
(20,60)	1.58; 94.0% of WMMSE	1.68	1.59	1.50	0.37	0.35
(20,100)	1.14; 92.7% of WMMSE	1.23	1.15	1.09	0.18	0.17

Extension # 1 / Mobile Users & Continuous Action Space

- Replace deep Q-learning by a deep actor-critic learning algorithm called deep deterministic policy gradient (DDPG) for continuous action space.
- Haas mobility model: travel between training episodes.



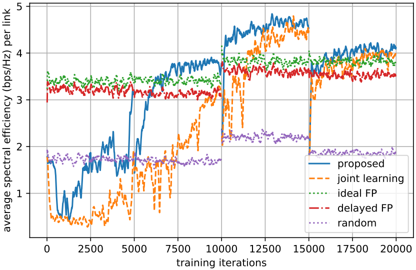


- Policy better experiences various device positions and interference conditions with mobility, so the performance consistently increases.

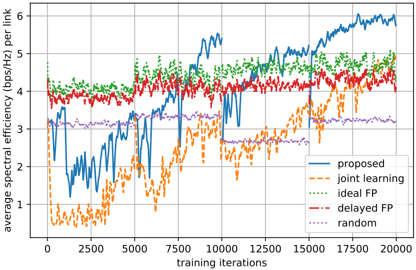
Extension # 2 / Subband Selection & Power Control.

- A link can be active on a single subband at a time with $p_{j,m}^{(t)} \leq P_{\max}$.
- [Tan, Zhang, and Liang '19] proposed an FP based solution for joint subband selection and power allocation.
- Joint DRL scheme's action set is the Cartesian product of available subbands and quantized transmit power levels.
- The computational complexity of FP and the action set complexity of joint DRL do not scale well for a large number of subbands.
- We propose a two-layer learning scheme, where
  - the top layer does discrete subband scheduling by deep Q-learning,
  - the bottom layer is responsible for continuous power allocation at the physical layer by DDPG

Extension # 2 / Simulation results / Training convergence



$M = 4$ subbands, $(K, N) = (5, 20)$.

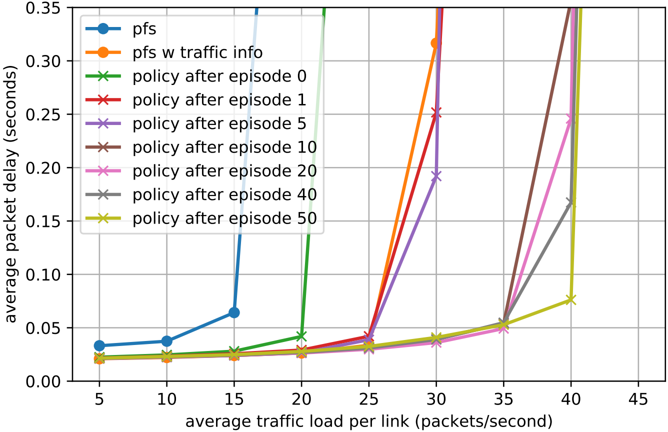


$M = 10$ subbands, $(K, N) = (10, 50)$.

Benchmarks:

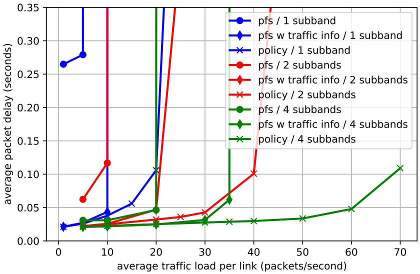
- Benchmarks:
  1. pfs: WMMSE (centralized and genie-aided with full instantaneous CSI) with user priorities adjusted to achieve proportional fairness.
  2. pfs with traffic information: WMMSE that enhances pfs' user priority assignment by also setting user priority to zero if user's queue is empty.

Testing the policy along the episodic training. $M = 1$.

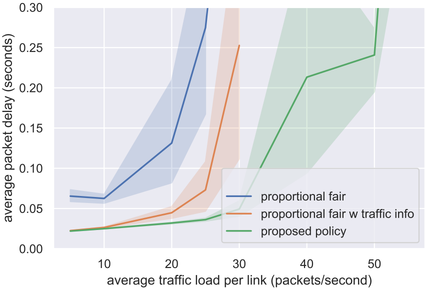


Policy is trained on $N = 5$ users on $K = 5$ cells, and tested on a larger deployment with $N = 20$ users on $K = 20$ cells.

Testing the policy on multiple subbands and seeds.

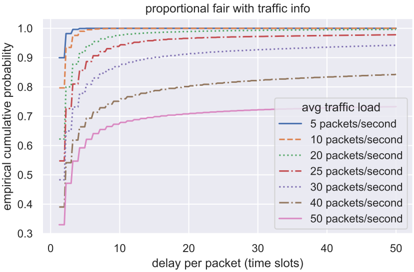


Test a converged policy on a $(N = 20 \text{ users}, K = 10 \text{ cells})$ scenario for total number of subbands $M \in \{1, 2, 4\}$.

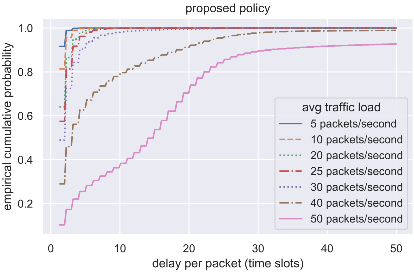


Testing a pre-trained policy on 10 different testing seeds. $N = 20$ links, $K = 10$ cells, $M = 2$ subbands.

CDF of all packet delays $(N, K, M) = (20, 10, 2)$.



proportional fair with traffic info.



proposed policy.

Summary

- A new distributed dynamic spectrum and power allocation algorithm based on deep reinforcement learning.
- The policy successfully maps traffic and channel states to physical resource allocations.
- User priorities connect physical layer resource management with network layer. Policy can achieve any traffic related network objective with a suitably designed reward function.
- Policy works well with delayed CSI and mismatched parameters.
- No need to produce a large amount of training data.
- In certain scenarios, the performance exceeds that of state-of-the-art algorithms WMMSE and FP. The distributed solution scales well.
- Available repositories:
  - `https://github.com/sinannasir/Power-Control-asilomar`
  - `https://github.com/sinannasir/Spectrum-Power-Allocation`

Future work on additional features:

- Multiple-input multiple-output (MIMO) beamforming: The challenge is the additional state-action complexity. Solution may involve a more sample-efficient DRL algorithm and a better neural network architecture or compressed parameters to reduce the complexity.
- User association: The distributed execution scheme needs to be modified. If user associations are not pre-determined, the agents should work above the base stations.

- **Thank you for your time, questions?**

Deep Q-learning algorithm

- Use a deep Q-network (DQN) parameterized by $\boldsymbol{\psi}$ to represent the Q-function values $q(\cdot, \cdot; \boldsymbol{\psi})$
- The optimal Q-function satisfies:
$$Q^{\pi^*}(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in S} \mathcal{P}^{a}_{ss'} \max_{a'} Q^{\pi^*}(s',a'),$$
where $\mathcal{R}(s,a) = \mathbb{E}\left[r^{(t+1)} \middle| s^{(t)} = s, a^{(t)} = a\right]$.
- It is an off-policy learning that stores experiences in a memory $\mathcal{D}$.
- For training, the mean-squared Bellman loss is defined as
$$L\left(\boldsymbol{\psi}, \mathcal{D}\right) = \mathbb{E}_{(s,a,r',s') \sim \mathcal{D}}\left[\left(y(r',s') - q\left(s,a;\boldsymbol{\psi}\right)\right)^2\right],$$
where the target is $y(r',s') = r' + \gamma \max_{a'} q\left(s',a';\boldsymbol{\psi}_{\text{target}}\right)$.
- $\boldsymbol{\psi}$ is updated using a stochastic gradient descent algorithm by
$$\nabla_{\boldsymbol{\psi}} \frac{1}{|\mathcal{B}|} \sum_{(s,a,r',s') \in \mathcal{B}} \left(y(r',s') - q\left(s,a;\boldsymbol{\psi}\right)\right)^2,$$
where the target is $y(r',s') = r' + \gamma \max_{a'} q\left(s',a';\boldsymbol{\psi}_{\text{target}}\right)$.

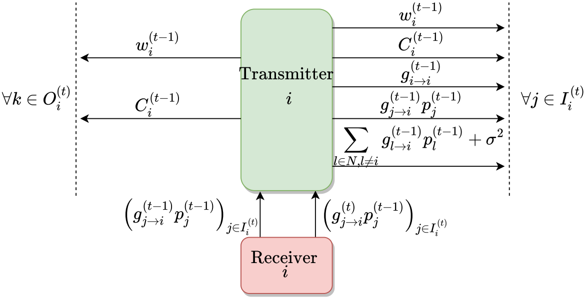
Local Information and Neighborhood Set

- *Interferer set* $I_n^{(t)}$: transmitters that cause interference at receiver n;

$$I_n^{(t)} = \left\{ i \in \mathcal{N}, i \neq n \middle| g_{i \to n}^{(t-1)} p_i^{(t-1)} > \eta\sigma^2 \right\}.$$

- *Interfered set*: $O_n^{(t)}$: links that suffer from transmitter n.

$$O_n^{(t)} = \left\{ k \in \mathcal{N}, k \neq n \middle| g_{n \to k}^{(t-1)} p_n^{(t-1)} > \eta\sigma^2 \right\}.$$



Preliminary for state set

- Regulated interferer and interfered neighborhood sets $\left(\bar{I}_n^{(t)}, \bar{O}_n^{(t)}\right)$.
- We set $\left|\bar{I}_n^{(t)}\right| = \left|\bar{O}_n^{(t)}\right| = c$.
  - Pick c-most significant interferer and interfered neighbors with following criteria:
    - the current received power from interferer $i \in I_n^{(t)}$ at receiver n,
    - the share of agent n on the interference at receiver $k \in O_n^{(t)}$.
  - If necessary, append virtual noise agents with an arbitrary negative weight and spectral efficiency. A virtual noise agent has zero downlink and interfering channel gains.

States

1. Local Information (7 inputs to DQN):

$$p_n^{(t-1)},\ 1/w_n^{(t)},\ C_n^{(t-1)},\ g_{n\to n}^{(t)},\ g_{n\to n}^{(t-1)},$$
$$\sum_{j\in N, j\neq n} g_{j\to n}^{(t)} p_j^{(t-1)} + \sigma^2,\ \sum_{j\in N, j\neq n} g_{j\to n}^{(t-1)} p_j^{(t-2)} + \sigma^2$$

2. From interfering neighbors (3 inputs each):

$$c \text{ interferers of current time slot: } g_{i\to n}^{(t)} p_i^{(t-1)},\ 1/w_i^{(t-1)},\ C_i^{(t-1)}, \quad \forall i \in \bar{I}_n^{(t)}$$

$$c \text{ interferers from history: } g_{i'\to n}^{(t-1)} p_{i'}^{(t-2)},\ 1/w_{i'}^{(t-2)},\ C_{i'}^{(t-2)}, \quad \forall i' \in \bar{I}_n^{(t-1)}$$

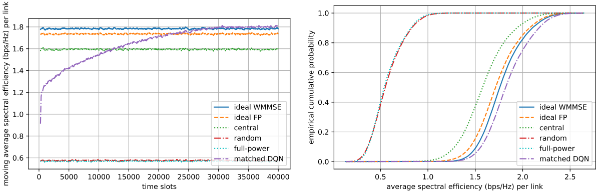
3. From interfered neighbors (4 inputs each):
   t'_n is the last time slot transmitter n was active,

$$g_{k\to k}^{(t-1)},\ 1/w_k^{(t-1)},\ C_k^{(t-1)},\ \frac{g_{n\to k}^{(t'_n)} p_n^{(t'_n)}}{\sum_{j\in N, j\neq k} g_{j\to k}^{(t-1)} p_j^{(t-1)} + \sigma^2}, \quad \forall k \in \bar{O}_n^{(t'_n)}.$$

Sum-rate maximization: multiple links per cell (IMAC)

Constraint $\sum_{j\in\mathcal{N}_k} p_{j,m}^{(t)} \leq P_{\max}$ becomes $p_{j,m}^{(t)} \leq P_{\max}$
$K = 19$ cells, M=1 subband, $R = 500$ m, $f_d = 10$ Hz.
(cross-link CSI is available to the DQN.)

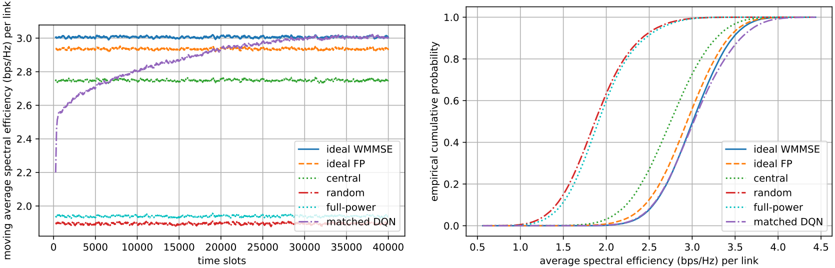
	average sum-rate in bps/Hz per link					
	DQN		benchmark power allocations			
links per cell	matched	unmatched	WMMSE	FP	central	full-power
2	1.84	1.58	1.78	1.74	1.59	0.57
4	1.25	1.06	1.24	1.22	1.10	0.25
random (1–4)	1.61	1.37	1.57	1.53	1.40	0.44



2 links per cell; (left) training (moving average of previous 250 slots); (right) testing.

Sum-rate maximization

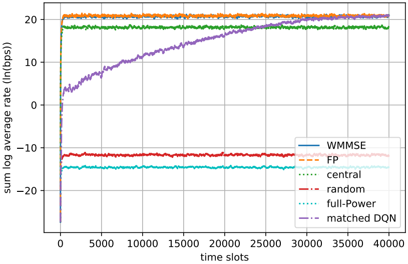
$N = 19$ links, $K = 19$ cells, M=1 subband, $R = 100$ m, $f_d = 10$ Hz. (cross-link CSI is available to the DQN.)

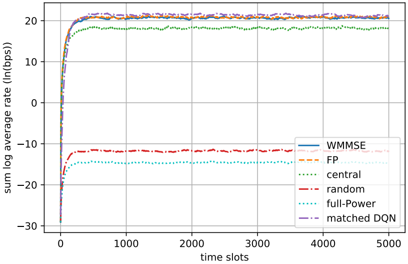


(left) training (moving average of previous 250 slots); (right) testing.

Proportionally fair scheduling

$N = 19$ links, $K = 19$ cells, M=1 subband, $R = 500$ m, $f_d = 10$ Hz.
(cross-link CSI is available to the DQN.)





(left) training; (right) testing.

Continuous Action Space

- [Men, Chen, Wu, and Cheng '19] showed that quantizing the action space with a logarithmic step size gives better outcomes than that of a linear step size for a different channel model.
- They proposed to replace deep Q-learning scheme by a deep actor-critic learning scheme called deep deterministic policy gradient (DDPG).
- Actor-critic learning trains an action-value function using a critic network, defined by ϕ;
- and uses this function estimate to train a policy parameterized by an actor network, defined by $\boldsymbol{\theta}$.
- Actor-critic learning is
  - as sample efficient as value based methods, and
  - as direct as policy based methods.

Actor-critic learning (deep deterministic policy gradient)

- The action is determined by $a = \mu(s; \boldsymbol{\theta})$ with policy parameters being $\boldsymbol{\theta}$.
- For exploration, a noise term can be added on the action values.
- The target policy μ^* satisfies the Bellman property:

$$Q^{\mu^*}(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in S} \mathcal{P}^a_{ss'} Q^{\mu^*}(s', \mu^*(a')),$$

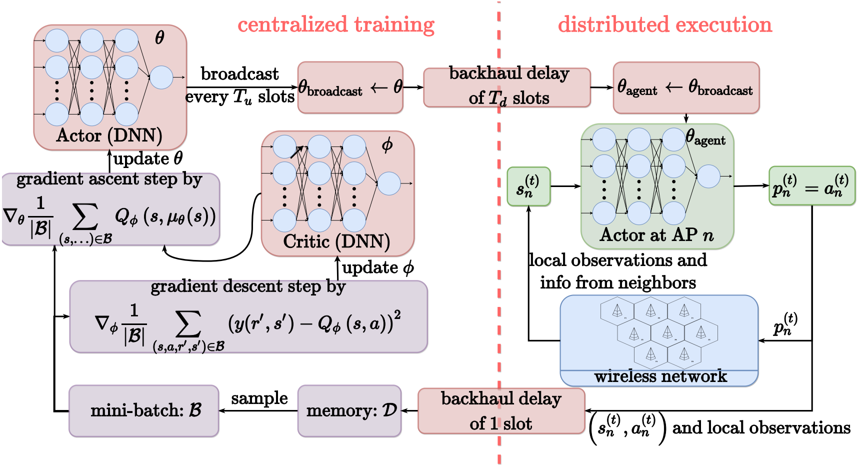
- Critic network is updated by

$$\nabla_{\boldsymbol{\phi}} \frac{1}{|\mathcal{B}|} \sum_{(s,a,r',s') \in \mathcal{B}} \left(y_{\text{critic}}(r', s') - q\left(s, a; \boldsymbol{\phi}\right)\right)^2,$$

  where $y_{\text{critic}}(r', s') = r' + \gamma q\left(s', \mu(s'; \boldsymbol{\theta}); \boldsymbol{\phi}_{\text{target}}\right)$.
- $q(s, a; \boldsymbol{\phi})$ is differentiable with respect to continuous action.
- The policy parameters are updated by the following gradient:

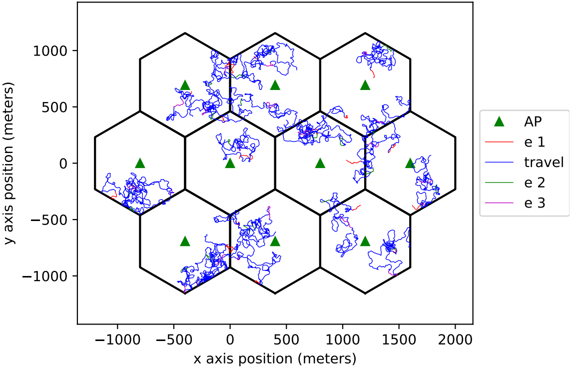
$$\nabla_{\boldsymbol{\theta}} \frac{1}{|\mathcal{B}|} \sum_{(s,\dots) \in \mathcal{B}} q\left(s, \mu(s; \boldsymbol{\theta}); \boldsymbol{\phi}\right).$$

DDPG based centralized training and distributed execution



Mobile Users / Training episodes and traveling

- Steady channel may cause overfitting to a certain network deployment.
- Haas mobility model: travel between training episodes.



- Correlation in Jakes model becomes $\rho_n^{(t)} = J_0(2\pi f_{d,n}^{(t)} T)$, $f_{d,n}^{(t)} = v_n^{(t)} f_c / c$;
- large scale-fading also varies with $\rho_{\mathrm{s},n}^{(t)} = e^{\frac{\Delta \mathbf{x}_n^{(t)}}{d_{\mathrm{cor}}}}$

Problem Formulation

- If link n selects subband m , we have $\alpha_{n,m}^{(t)} = 1$ and $\alpha_{n,j}^{(t)} = 0, \forall j \neq m$.
- SINR at receiver n on subband m in time slot t:

$$\gamma_{n,m}^{(t)} = \frac{\alpha_{n,m}^{(t)} g_{n \to n,m}^{(t)} p_n^{(t)}}{\sum_{l \neq n} \alpha_{l,m}^{(t)} g_{l \to n,m}^{(t)} p_l^{(t)} + \sigma^2},$$

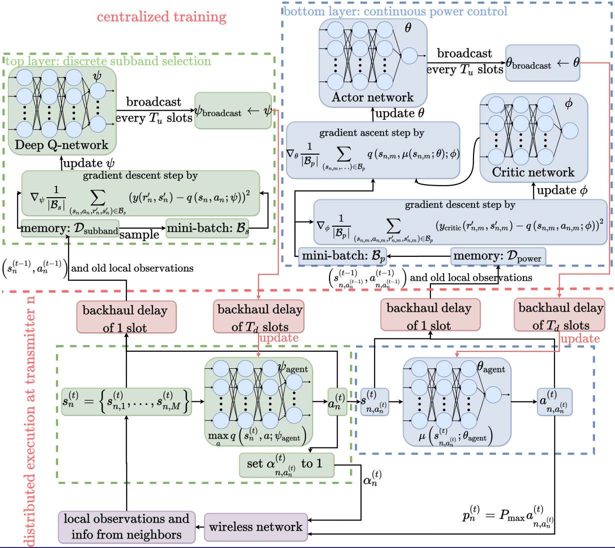
- Spectral efficiency:

$$C_n^{(t)} = \sum_{m=1}^{M} C_{n,m}^{(t)} = \sum_{m=1}^{M} \log\left(1 + \gamma_{n,m}^{(t)}\right).$$

- Let $\boldsymbol{\alpha}^{(t)} = \left[\alpha_{1,1}^{(t)}, \alpha_{1,2}^{(t)}, \dots, \alpha_{N,M}^{(t)}\right]^{\intercal}$ and $\boldsymbol{p}^{(t)} = \left[p_1^{(t)}, \dots, p_N^{(t)}\right]^{\intercal}$, the optimization problem in slot t:

$$\boxed{\begin{aligned} \underset{\boldsymbol{p}^{(t)}, \boldsymbol{\alpha}^{(t)}}{\text{maximize}} \quad & \sum_{n=1}^{N} C_n^{(t)} \\ \text{subject to} \quad & 0 \leq p_n^{(t)} \leq P_{\max}, \forall n \in \mathcal{N}, \\ & \alpha_{n,m}^{(t)} \in \{0,1\}, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \\ & \sum_{m \in \mathcal{M}} \alpha_{n,m}^{(t)}, \forall n \in \mathcal{N}, \end{aligned}}$$

DDPG based centralized training and distributed execution



Extension # 2 / Simulation results / Testing performance

(K, N) (cells, links)	M subbands	average sum-rate performance in bps/Hz per link: reinforcement learning proposed	joint	other schemes: ideal FP	delayed FP	random	output layer size: reinforcement learning proposed	joint	average iterations FP
$(5, 20)$	1	1.51	1.50	1.58	1.46	0.41	1 + 1	10	70.30
	2	2.63	2.64	2.66	2.46	0.99	2 + 1	20	102.08
	4	4.57	4.38	3.81	3.57	2.12	4 + 1	40	122.15
$(10, 50)$	1	1.26	1.26	1.31	1.21	0.25	1 + 1	10	72.83
	2	2.08	2.10	2.08	1.92	0.59	2 + 1	20	96.32
	4	3.34	3.34	2.90	2.68	1.31	4 + 1	40	185.93
	5	3.79	3.76	3.18	2.94	1.64	5 + 1	50	206.38
	10	5.71	4.41	4.44	4.08	2.99	10 + 1	100	287.70

- Results show that a pretrained policy is still usable on new deployments and the proposed approach is better scalable than the benchmarks.

Pseudo-code for distributed execution.

Parameters: ϵ-greedy algorithm's ϵ.
Inputs: Deep Q-network parameters at agents ψ_{agent}.
Distributed execution $(\boldsymbol{\psi_{\text{agent}}})$ for time slot t:
for agent $n = 1, 2, \ldots, N$ **do**
 Agent n observes its local environment and uses information from its neighbors to form its current local state $s_n^{(t)}$.
 Agent sets its current action to $a_n^{(t)} = \arg\max_a q\left(s_n^{(t)}, a; \psi_{\text{agent}}\right)$ using deep Q-network with parameters ψ_{agent}.
 If index n is divisible by $t \mod N$, apply ϵ-greedy strategy for exploration during training and agent replaces $a_n^{(t)}$ with a random action with a probability of ϵ.
 Agent translates its action to its allocation decision, i.e., $\left[p_{n,1}^{(t)}, \ldots, p_{n,M}^{(t)}\right]^{\mathsf{T}}$, after power auction at base station b_n.
end for

Output: $\boldsymbol{p}_m^{(t)}$, $\forall m \in \mathcal{M}$, and state-action pairs $\left(s_n^{(t)}, a_n^{(t)}\right) \forall n \in \mathcal{N}$.

Pseudo-code for centralized training.

Parameters: Learning rate λ_{lr}.

Inputs:

Deep Q-network parameters ψ, $\psi_{\mathrm{broadcast}}$, ψ_{agent}.

Global memory $\mathcal{D}_{\mathrm{g}}$ & experience-replay memory of the current episode $\mathcal{D}$.

Centralized training $(\psi, \psi_{\mathbf{broadcast}}, \psi_{\mathbf{agent}}, \mathcal{D}_{\mathbf{g}}, \mathcal{D})$**:**

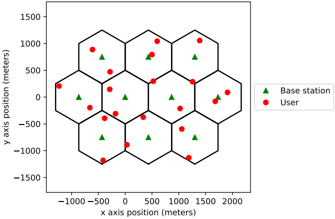
Randomly sample a mini-batch $\mathcal{B}$ from the experiences in $\mathcal{D}_{\mathrm{g}}$ and $\mathcal{D}$.

Update the parameters ψ using a gradient descent step with learning rate equal to λ_{lr} and the gradient $\nabla_{\boldsymbol{\psi}} \frac{1}{|\mathcal{B}|} \sum_{(s,a,r',s') \in \mathcal{B}} \left(y(r', s') - q\left(s, a; \boldsymbol{\psi}\right)\right)^2$.

If it has been T_u since last policy broadcast, update $\psi_{\mathrm{broadcast}}$ by ψ and initiate a broadcast process which will take T_d time slots. At the end of the broadcast process, ψ_{agent} will be set to $\psi_{\mathrm{broadcast}}$.

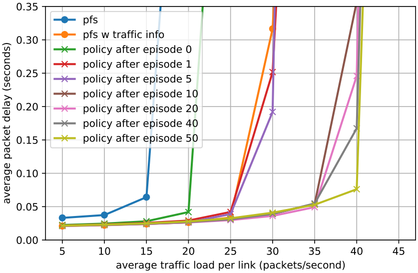
Output: Updated deep Q-network parameters ψ, $\psi_{\mathrm{broadcast}}$, ψ_{agent}.

Simulation Setup

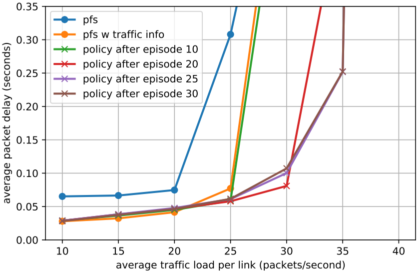


maximum transmit power $P_{\max}$	23 dBm
subband bandwidth	10 MHz
number of subbands	1 to 4
slot duration T	20 ms
traffic arrivals	Poisson arrivals / 500 Kbits
path loss (in dB)	$120.9 + 37.6 \log_{10}(d)$
shadowing standard deviation	8 dB
AWGN power	-114 dBm
maximum Doppler frequency	10 Hz

Testing the policy along the episodic training. $M = 1$.

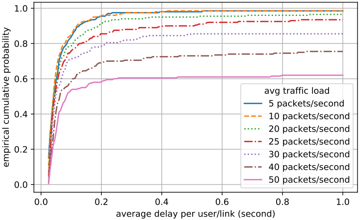


Policy is trained on $N = 5$ users on $K = 5$ cells, and tested on a larger deployment with $N = 20$ users on $K = 20$ cells.

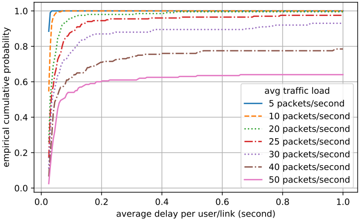


Policy is trained and tested on $N = 10$ users on $K = 5$ cells.

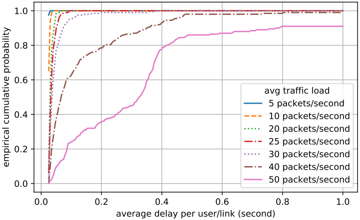
CDF of average user delay $(N, K, M) = (20, 10, 2)$.



(a) proportional fair



(b) proportional fair with traffic info



(c) proposed policy

Some other side problems:

- Better and easily tunable training and exploration schemes to better adapt to the environment non-stationarity of the multi-agent setting.
- We simplified the state set design, but its design can be improved by analyzing the hidden-layer weights of a trained policy that uses global CSI and picking the environment features that impact the decision strategy most strongly.