Reinforcement Learning for Power Allocation in Multi-User Wireless Communication Systems

Asma Boukhdhir, Aymen Daoud, and Farouk Nasr
Dep. of Computer and Software Engineering, Polytechnique Montréal
Montréal, Canada
December 20, 2024

***Abstract*—Wireless communication systems, such as cellular networks, face an ongoing challenge in managing limited resources like power to ensure high-quality service for multiple users. With the rapid growth of devices and applications, resource allocation strategies must dynamically adapt to fluctuating demands, interference patterns, and environmental conditions. In wireless networks, efficient power allocation becomes critical as interference management directly impacts user performance. Traditional rule-based approaches to power allocation, such as water-filling or fixed policies, assume static or semi-static conditions and struggle to adapt to the dynamic nature of wireless environments. To address these limitations, we propose the application of Deep Reinforcement Learning (DRL) methods. Among DRL algorithms, the Deep Q-Network (DQN) has demonstrated notable success in handling decision-making problems in highdimensional state spaces, offering a promising approach for dynamic power allocation.**

I. Problem Statement

In multi-user wireless networks with multiple base stations (BSs), users experience interference from adjacent cells and competing users. Balancing the power allocation to maximize network performance while minimizing interference is a significant challenge. For instance:

- Increasing power to one user enhances its quality of service (QoS) but may degrade the QoS of neighboring users due to increased interference.
- Dynamic conditions such as user mobility, varying channel quality, and unpredictable interference further complicate the problem.

Existing solutions, including model-based optimization and heuristic algorithms, often fail to generalize well in highly dynamic environments. This project addresses the need for a robust, adaptive, and scalable power allocation strategy using reinforcement learning (RL) algorithms. The RL agent will be trained to learn optimal power allocation policies that can dynamically adjust to real-time changes in network conditions.

II. Objectives

The objectives of this project are as follows:

1) To develop and implement reinforcement learning algorithms for learning optimal power allocation strategies in multi-user wireless networks.
2) To compare the performance of the RL-based approach against traditional power allocation techniques, such as equal power allocation, maximum power allocation, and random allocation.
3) To demonstrate the adaptability of the RL model in highly dynamic wireless environments.

III. Methodology

The project will involve designing a reinforcement learning framework to model the power allocation problem as a Markov Decision Process (MDP).

A. Environment Setup

The environment is a multi-cell massive MIMO (Multiple Input Multiple Output) environment for realistically simulating wireless communication systems.It is designed with the following parameters:

- **N:** Number of cells or base stations (BSs).
- **M:** Number of transmission antennas per BS.
- **K:** Number of UEs (User Equipments) per BS .
- **Bandwidth (BW):** Fixed at 10 MHz.
- **Noise Figure (NF):** 7 dBm.
- **Power Constraints:** Transmission power levels range from -20 dBm to 23 dBm.
- **Channel Models:** Includes Rayleigh fading, shadowing, and path loss.

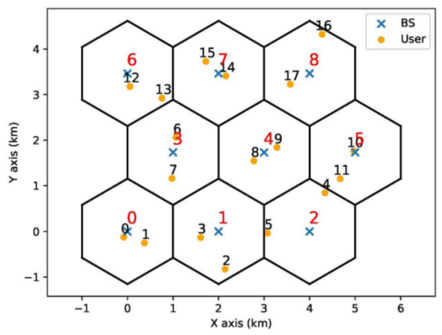

Figure 1: An illustrative example of a multi-user cellular network with 9 cells. In each cell ,a BS serves 2 users simultaneously

B. Reinforcement Learning Setup

In communication networks, each BS-user link can be regarded as an agent and thus a multi-agent system is studied. However, multi-agent training is difficult since it needs much more learning data, training time and deep neural networks parameters. Therefore , centralized training is considered in this project, and only **one agent** is trained to allocate the same power level to each user for each BS.

- **State Space:**
  The state space is **continuous** and theoretically **unbounded**, representing a global observation of the entire network. It is a function of time and is **not tied to a specific base station (BS) or user equipment (UE)**, aligning with the overall optimization objective.
  1) **Current Transmission Power:**
     - Represents the amount of power each base station (BS) uses to transmit signals to users.
     - Typically measured in *decibels (dB)*, it determines the **strength** and **range** of the transmitted signal.
  2) **Average Signal-to-Interference-plus-Noise Ratio (SINR):**
     - The *SINR* quantifies the quality of the received signal and is defined as:
       $$\text{SINR} = \frac{\text{Signal Power}}{\text{Interference Power} + \text{Noise Power}}$$
     - Higher SINR values indicate better signal quality, which leads to improved communication performance.
     - To account for all users served by each base station, the **average SINR** is computed across all users in the network.
  3) **Total Sum-Rate:**
     - The total sum-rate measures the **maximum achievable data rate** (in bits per second per Hz) across all base stations (BSs) and user equipment (UEs).
     - It reflects the *spectral efficiency* of the network and is calculated as:
       $$R_{\text{sum}} = \sum_{n=1}^{N}\sum_{k=1}^{K}\log_2\left(1 + \text{SINR}_{n,k}\right)$$
       where $\text{SINR}_{n,k}$ is the SINR for user k in cell n.
     - This metric provides an overall measure of how efficiently the available bandwidth is utilized.
- **Action Space:**
  The action space is composed of 10 discrete power levels. The level 0 reflects no transmission between between the BS and the user, while the remaining 9 levels represent power values equally distributed between P_{min} = -20dBm and P_{max} = 23dBm. At each time step, the agent selects a power level from this set to be allocated.

- **Objective function:**
  The optimization target is to maximize this generic sum-rate objective function under maximum power constraint, and it is formulated as :
  $$\max_{p^t} R^t_{\text{sum}} \quad \text{subject to} \quad 0 \le p^t \le P_{\text{max}}$$
  where R^t_{sum} is the total sum-rate at time step t.

- **Reward function:** The reward is designed to give feedback to the RL agent, guiding its decisions and learning process.It is defined with respect to the objective function as **the total sum rate** R^t_{sum}.

IV. Environment Simulation

In wireless communication systems, such as MIMO (Multiple-Input Multiple-Output), two key matrices are used to model and optimize signal transmission: the channel matrix and the precoding matrix.

1) **The Channel Matrix:**
   A *channel* describes how signals travel from the base station (BS) antennas to the users (UEs). This propagation is influenced by external factors, leading to the following phenomena:
   - **Fading:** Weakening of the signal strength.
   - **Reflection:** Bouncing off buildings and other large obstacles.
   - **Scattering:** Spreading out of the signal due to smaller obstacles.

   These factors affect both **the strength and direction** of the transmitted signal.

   The channel matrix is therefore used to model two prominent effects:
   a) **Large-Scale Fading:** Accounts for slow changes in signal strength caused by the distance between the user and the base station (BS), as well as obstacles like buildings or terrain.
   b) **Small-Scale Fading:** Models rapid variations in signal strength due to reflections and scattering effects.
2) **The Precoding matrix:** To ensure reliable and high-quality communication, *precoding* is applied. Precoding is a signal processing technique that optimizes signal transmission from the base station to multiple users by addressing interference. Concretely, precoding ensures :
   - Directing signals to the intended users so that they constructively combine at their receivers.
   - Minimizing inter-user interference to improve signal quality.

   In this system, we use *Zero-Forcing (ZF) precoding*, a widely adopted method that eliminates inter-user interference by ensuring that the signals are orthogonal at the receivers.
   Mathematically, ZF precoding solves for a combining matrix that "forces" the interference to zero.

V. Algorithms Description

In this project, various deep reinforcement learning (DRL) algorithms were implemented and evaluated to compare their effectiveness in learning optimal policies for complex, model-free power allocation (PA). The selection of algorithms was carefully guided by the problem's specific characteristics, namely a **continuous state space** and a **discrete action set**. We have also included **value-based methods** as well as **policy-based ones**.

The performance of these DRL algorithms was benchmarked against traditional power allocation techniques, which serve as baseline methods for comparison.

1) Traditional algorithms:
   - Random Power allocation: Allocate power to users randomly within the allowed range $[P_{\min}, P_{\max}]$.
   - Equal Power allocation: Allocate the same amount of power to all users in each cell $P_{\max}/k$.
   - Allocate the maximum allowable power to all users in each cell $P_{\max}$.
2) Reinforcement learning algorithms:
   Value-based methods:
   - **DQN** : Deep Q-Learning is an extension of the basic Q-Learning algorithm, which uses deep neural networks to approximate the Q-values.It is used to address the curse of dimensionality and the lack of generalization. This algorithm is used in conjuction with **experience replay** which tackles the correlation between sequential data samples, stabilizing the learning and enhancing the data efficiency. Additionally, the Q-Network is optimized towards a **frozen target network** that is periodically updated with the latest weights every fixed number of steps. This makes training more stable by preventing short-term oscillations from a moving target.
   - **DDQN** : The DQN algorithm is widely recognized for its tendency to exhibit significant overestimation bias. To address this limiation, the double DQN(DDQN) algorithm was introduced as an adaptation of DQN. By mitigating overestimations, DDQN achieves improved overall performance. This enhancement is accomplished by decoupling the task of selecting the optimal action to take at each time step and estimating the value of this action in the current state. The selection is done by the target network while the value network is used for evaluation.
   - **Rainbow DQN** : The Rainbow DQN algorithm adds six improvements to the vanilla implementation of DQN. These optimizations have proved to achieve state-of-the-art performance in some benchmarks, including the Atari 2600 game. Therefore, this advanced implementation of deep Q-learning is worth testing in learning optimal policies for the PA problem. In this project 5 of the 6 optimizations are used:
     - **Double Q-Learning**: Reduces overestimation bias by separating the action selection and evaluation processes.
     - **Dueling Network Architecture**: Enhances the ability to differentiate between the value of a state and the advantage of actions.
     - **Prioritized Experience Replay**: Samples more important experiences more frequently rather than random sampling. The priority assigned to each data sample corresponds to the TD error achieved at the state-action pair.
     - **Noisy Networks**: Introduces adaptive exploration by adding stochastic noise to the network weight rather than relying solely on the ϵ-greedy policy.
     - **Multi-step Learning**: Propagates rewards over multiple steps to better capture long-term dependencies therefore n-step TD error is used instead of TD(0).

Policy-based methods:

- **Proximal Policy Optimization (PPO)** : a natural policy algorithm and one of the most widely used and efficient algorithms for training agents in reinforcement learning due to its simplicity, stability, and performance. Its core idea consists in optimizing a "surrogate" objective function to ensure that policy updates are small and controlled, preventing drastic changes that could destabilize learning.
  The algorithm improves upon earlier policy gradient methods like Trust Region Policy Optimization (TRPO) by simplifying the trust region constraint, making it easier to implement and computationally efficient.
- **PPO with Generalized Advantage Estimation (GAE)**
  Generalized Advantage Estimation (GAE) is a technique used in reinforcement learning (RL) to estimate the advantage function more effectively by reducing the variance while maintaining a manageable bias. It was introduced by John Schulman in the context of the Trust Region Policy Optimization (TRPO) algorithm but is also widely used in other policy gradient methods like Proximal Policy Optimization (PPO).

  KEY CONCEPTS

  Advantage Function

  The advantage function $A(s_t, a_t)$ quantifies how much better (or worse) taking a specific action a_t in state s_t is compared to the average action in that state. Formally:

  $$A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$$

  where:
  - $Q(s_t, a_t)$: Action-value function.
  - $V(s_t)$: State-value function.

  Traditional Advantage Estimation

  Using a single-step or n-step return for $A(s_t, a_t)$ can result in high variance or bias.

  GAE

  GAE introduces a trade-off between bias and variance by combining multi-step advantage estimates in a weighted manner. The key idea is to use exponentially-weighted moving averages of the temporal difference (TD) residuals.

GAE Formula

GAE computes the advantage estimate as:

$$\hat{A}_t^{\text{GAE}} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

where:

- γ: Discount factor (controls how much future rewards are considered).
- λ: GAE parameter (controls the bias-variance trade-off).
- δ_t: Temporal difference residual at time t, computed as:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

The advantage is essentially a weighted sum of these TD errors, with the weight determined by $\gamma\lambda$.

VI. Performance evaluation

The performance of RL algorithms will be evaluated in terms of **convergence, stability and average downlink rate achieved after convergence.**

The simulation was done using N=7,M=32 and K=10.

A. DQN Training Analysis

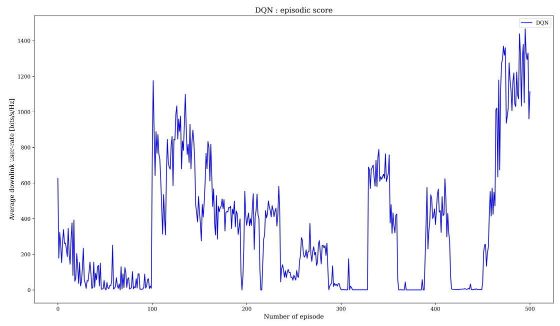

Figure 2: DQN Training Plot

The DQN episodic score plot shows highly variable performance across the 500 episodes. Initially, the agent struggles to achieve high rewards, with noticeable fluctuations due to the exploration phase governed by the epsilon-greedy policy. In the middle phase (between episode 100 and episode 450), the agent begins to show improvements but experiences periods of inconsistency, likely due to overestimation bias or suboptimal exploration-exploitation balance. In this phase, there is no clear upward trend and significant drops of performances, showcasing high training instability.Following these drops,the agent has nearly zero performance, but recovers and oscillates in later stages. This pattern suggests the need for more robust stabilization methods (e.g., smaller learning rates, prioritized experience replay) to ensure consistent learning and prevent performance collapses reflecting catastrophic frogetting. In the final stage, the DQN agent successfully showcases convergence through gradual improvement. This convergence begins near the end of the training process (approximately after **450 episodes**) which showcases the slow convergence of the DQN algorithm. The final reward is around **1200**.

B. DDQN Training Analysis

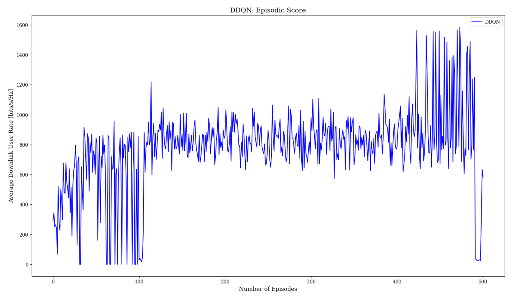

Figure 3: DDQN Training Plot

The DDQN episodic score plot demonstrates the agent's learning behavior over 500 episodes. Initially, the agent exhibits considerable fluctuations with rewards varying between 0 and 800 although on average it showcases a consistent upward trend. After the episode 100, the training stabilizes and the agent successfully converges to a quite good policy. Compared to DQN, DDQN shows improved long-term stability and reduced overestimation bias, as evidenced by the smoother and more sustained performance improvements in later episodes. The DDQN as shown in the graph has a more steady performance without drops (zero performance) like experienced in the DQN training plot. The convergence is much faster and the final reward is clearly higher than 1200 achieved by the DQN agent.

C. Rainbow DQN Training Analysis

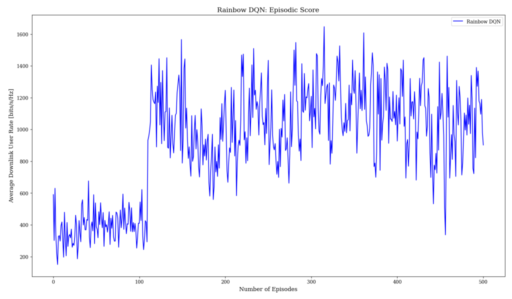

Figure 4: PPO Training Plot

The plot shows the average dowlink rate achieved by the rainbow DQN which is an advanced implementation of the DQN algorithm. The reward starts relatively low, fluctuating between 200 and 600. This phase corresponds to the exploration stage, where the Rainbow DQN algorithm is learning and improving its policy. Around episode 100, there is a sharp improvement in performance as the reward jumps to 1000.This sudden improvement suggests that the agent has successfully started learning a better policy and is exploiting the environment more effectively. Finally, the performance converges and remains relatively steady with continued high rewards but some occasional sharp drops,likely due to exploration. Nonetheless, the general trend remains stable,confirming that the agent has learned a good policy with a reward of approximately 1400. To conclude, the rainbow DQN effectively shows faster convergence and better overall performance compared to the vanilla implementation of DGN.

D. DQN vs Rainbow DQN and Power Allocation Algorithms

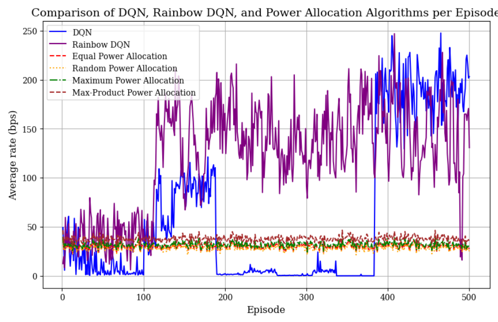

Figure 5: DQN vs Rainbow DQN and Power Allocation Algorithms comparison

The comparison plot highlights the performance of the DQN-based power allocation algorithms during training ,including the vanilla implementation of DQN and the Rainbow DQN, versus heuristic-based algorithms (Equal, Random, Maximum, and Max-Product Power Allocation) evaluated over 500 episodes. Initially, the DQN's performance is highly unstable, fluctuating significantly and often underperforming compared to the heuristic methods. Around episode 400, the DQN begins to improve dramatically, surpassing all the baseline algorithms and achieving a significantly higher average rate. This indicates that the DQN learns an optimized policy over time that outperforms static and probabilistic strategies. On the other hand,the optimized implementation of Rainbow DQN consistently surpasses the baselines after the episode 100. The heuristic methods, while consistent throughout the training process, demonstrate their limitations in achieving peak performance compared to the learned policies of deep RL agent.

E. PPO with GAE Training Performance Analysis

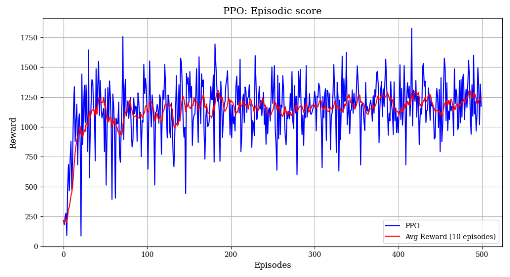

Figure 6: PPO Training Plot

The PPO Training Performance plot shows the episodic rewards (blue) and a rolling average reward over 10 episodes (red). The graph shows that the PPO algorithm performs well, with rewards increasing rapidly in the early episodes and stabilizing around 1200–1300 as training progresses. The algorithm converges in far less than 100 episodes which demonstrates the strength of this algorithm. It also achieves very high performance from time to time with peaks of approximately 1750.The final average is also quite high.

F. Rainbow DQN vs PPO and Power Allocation Algorithms Analysis

After training different RL agents using DQN-based algorithms and policy gradient methods, we will compare the best performing DQN implementation: Rainbow DQN and PPO combined with GAE optimization. The figure showcases the evaluation reward of both algorithms after the whole training process, the evaluation is done on 200 epsiodes to ensure the reliability of the results by masking the randomness effect.

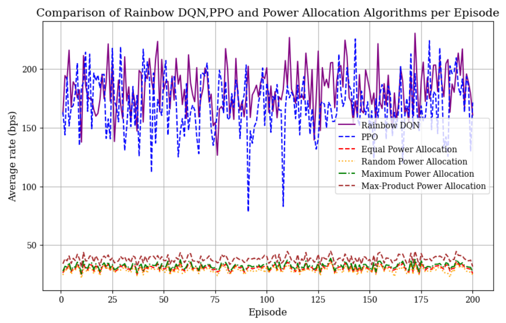

Figure 7: DQN Comparison

Algorithm	**Average downlink rate (Gbps)**
Rainbow DQN	182.39
PPO	169.78
Equal Power Allocation	30.55
Random Power Allocation	28.67
Maximum Power Allocation	32.06
Max-Product Power Allocation	38.14

Table I: Average downlink rate across 200 evaluation episodes

Based on the graph and the above table, both Rainbow DQN and PPO significantly outperform all power allocation algorithms, showcasing the advantage of reinforcement learning in dynamic environments whereas the benchmarking algorithms (equal, random, maximum, and max-product power allocation) maintain relatively stable but lower average rates . The gap is significantly high as the RL algorithms achieves an average score ≈ 170 Gbps which is higher than 10 times the score achieved by the naive PA algorithms ≈ 30 Gbps. Additionally, the rainbow DQN and PPO exhibits some fluctuations in performance with PPO demonstrating higher variability with lower drops. Based on the table, both algorithms achieves comparable average downlink rates. Though, the rainbow DQN surpasses the PPO with an average downlink rate of 182 Gbps.

VII. Analysis of results

Key Findings

- DQN,DDQN,Rainbow DQN and PPO demonstrated good performance in learning optimized power allocation policies while having different caracteristics(convergence, sample efficiency, and stability)
- Optimizing the deep RL algorithms with different tricks like those in Rainbow DQN which are added to the vanilla implementation of DQN can considerably enhance the performance.
- The highest overall performance for the specific hyperparameters we used corresponds to the Rainbow DQN, whereas PPO has a very comparable performance.

VIII. Conclusion

Through this project, we successfully applied deep reinforcement learning (DRL) algorithms to address a real-world challenge in the telecommunications domain. Specifically, we tackled the complex issue of power allocation in wireless communication networks. Our work demonstrated the capability of DRL models to learn and implement optimized policies for power allocation, delivering significantly improved network performance compared to naive algorithms. This was validated using various key metrics, including throughput and Signal-to-Interference-plus-Noise Ratio, showcasing the potential of DRL in enhancing network quality. Through the course of this project, we were able to better understand the key ideas behind different popular RL algorithms, their advantages, limitations as well as their adaptability in complex and highly dynamic environments. In this project, we have also successfully simulated an MDP environment, more concretely a multi-user network or MIMO system. We aimed to realistically model the environment while maintaining simplicity, avoiding unnecessary complexity in the design. To achieve this, we drew inspiration from prior research, particularly the study titled ”Power Allocation in Multi-User Cellular Networks: Deep Reinforcement learning Approaches” [1], which served as a foundational reference for our approach.Finally, it is important to acknowledge that fine-tuning the hyperparameters of the models could potentially enhance performance further. However, this aspect was beyond the scope of our current work and was not pursued.

Here is the link to the Google Colab notebook: RL project Notebook

Note: ChatGPT was used to help improve this report’s grammar, sentence structure, and overall syntax.

IX. REFERENCES

[1]L. Sanguinetti, A. Zappone and M. Debbah, ”Deep Learning Power Allocation in Massive MIMO,” 2018 52nd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 2018, pp. 1257-1261, doi: 10.1109/ACSSC.2018.8645343. keywords: Resource management;Training;MIMO communication;Complexity theory;Channel estimation;Artificial neural networks;Deep learning,