CS258 Final Report: The RSA Problem

Anurag Deotale
Group 2

I. Methods: RL-based Routing

A. RL Algorithms

To solve the Routing and Spectrum Allocation (RSA) problem in optical networks, we employed two reinforcement learning (RL) algorithms: Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN).

Proximal Policy Optimization (PPO): PPO is a policy gradient method designed for stable and efficient training. It uses an actor-critic framework where the actor updates the policy based on the critic's evaluation. PPO employs a clipped objective function that prevents large updates, ensuring stable learning. This clipping mechanism balances exploration and exploitation, making PPO suitable for complex tasks like RSA.

Deep Q-Networks (DQN): DQN is a value-based RL algorithm that approximates the Q-values using a deep neural network. The Q-values represent the expected future rewards for taking a particular action in a given state. DQN uses experience replay, where experiences are stored in a buffer and sampled randomly during training to break correlations. Additionally, DQN utilizes target networks to stabilize training by keeping a separate network for generating target Q-values.

B. State Space

The state space in our RL environment represents the current utilization of the network's links. It provides the RL agent with the necessary information to make informed routing and spectrum allocation decisions. The state space is defined as a dictionary with a single key "utilization", mapping to a vector of normalized utilization values for each link in the network. Formally, the state space is expressed as:

$$\text{utilization} = \left[\frac{o(e_1)}{c(e_1)}, \frac{o(e_2)}{c(e_2)}, \ldots, \frac{o(e_n)}{c(e_n)}\right]$$

where $o(e)$ denotes the number of occupied slots on link e, and $c(e)$ denotes the capacity of link e. This representation ensures that the agent perceives the current load on each link, facilitating efficient decision-making.

C. Action Space

The action space in our environment defines the set of possible actions the RL agent can take at any given time. Each action corresponds to the allocation of a spectrum slot on a link. Given the discrete nature of spectrum allocation, the action space is represented as a discrete set of actions, where each action corresponds to a specific spectrum slot. Formally, if the capacity of each link is C, the action space is defined as:

$$\mathcal{A} = \{0, 1, 2, \ldots, C - 1\}$$

In our simulations, the link capacity C is set to 10, resulting in an action space with 10 discrete actions. Each action represents an attempt to allocate a specific slot on the chosen link.

D. Reward Function

The reward function in our RL environment is designed to guide the agent towards efficient routing and spectrum allocation. It provides positive feedback for successful allocations and negative feedback for blocked requests. Additionally, it incorporates a penalty based on link utilization to encourage the agent to use network resources efficiently. The reward function is defined as follows:

$$R = \begin{cases} 1 - \text{avg\_utilization} & \text{if request is allocated} \\ -10 - \text{avg\_utilization} & \text{if request is blocked} \end{cases}$$

where avg_utilization is the average utilization of all links in the network, given by:

$$\text{avg\_utilization} = \frac{1}{|E|} \sum_{e \in E} \frac{o(e)}{c(e)}$$

This design ensures that the agent receives a higher reward for successful allocations with lower overall utilization, promoting efficient use of network resources.

E. Starting State

Each episode begins with an initial state where all link utilizations are zero, indicating no active requests. This starting state provides a clean slate for the agent to begin learning the optimal routing strategies.

F. Episode Termination

An episode terminates after a fixed number of requests have been processed, specifically 100 requests in our experiments. The termination condition is met when all requests are either routed successfully or blocked. This episodic structure allows the agent to learn from a series of routing decisions and their outcomes.

II. Heuristic Algorithm

In addition to RL-based methods, we implemented a heuristic algorithm to serve as a baseline for comparison. The heuristic algorithm uses a greedy spectrum allocation approach, which operates as follows:

- **Shortest Path Calculation**: For each request, calculate the shortest path from the source to the destination using Dijkstra's algorithm.
- **Spectrum Allocation**: Attempt to allocate spectrum slots along the path using the smallest available index. This greedy approach minimizes the likelihood of blocking by preferring lower-indexed slots.
- **Blocking Condition**: If no spectrum slot is available on any link along the path, the request is blocked.
- **Reward and Objective Calculation**: Calculate the total rewards and the objective value based on the final utilization of the network. The total reward is the sum of rewards for all requests, and the objective value is the average link utilization across the network.

III. Method: Spectrum Allocation

For spectrum allocation, we employed a heuristic-based approach inspired by the index-based allocation method used in previous work. The strategy involves allocating the smallest available spectrum slot on the path from the source to the destination. This heuristic provides a simple yet effective way to manage spectrum resources across the network. The spectrum allocation process is as follows:

1) **Shortest Path Calculation**: For each incoming request, the shortest path from the source to the destination is calculated using Dijkstra's algorithm.
2) **Slot Allocation**: Starting from the first link on the path, the algorithm checks for the availability of spectrum slots. The smallest available slot is allocated. If any link along the path cannot accommodate the request due to full capacity, the request is blocked.
3) **Update Utilization**: Once a slot is allocated, the utilization of the corresponding links is updated accordingly.

This heuristic ensures that the network can handle requests efficiently while maintaining a simple allocation mechanism.

IV. Results

The performance of the RL-based routing methods (PPO and DQN) and the heuristic algorithm was evaluated based on the rewards and the objective value. The results demonstrate the effectiveness of RL algorithms in optimizing network utilization.

A. Learning Curves

The learning curves for PPO and DQN show the rewards over episodes, indicating the improvement in routing strategies over time. The RL algorithms effectively learn to minimize blocking and maximize successful routing.

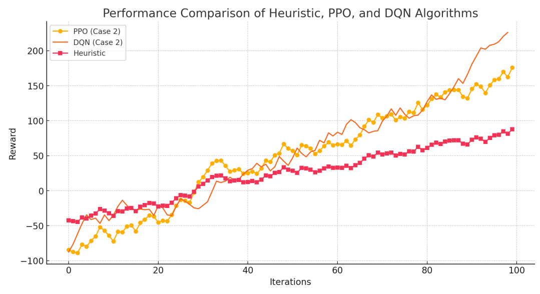

Fig. 1. Performance comparison of Heuristic, PPO, and DQN algorithms (Case 1).

Fig.1 shows the performance comparison of Heuristic, PPO, and DQN algorithms for Case 1. It demonstrates how the rewards evolve over iterations, highlighting the efficiency of RL methods in learning optimal routing strategies.

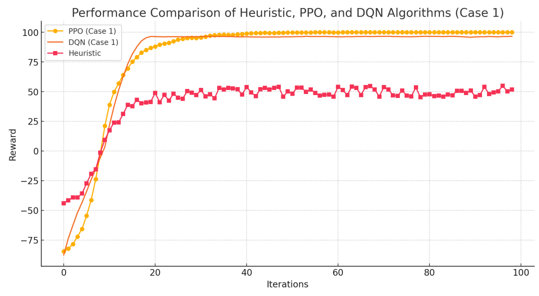

Fig. 2. Performance comparison of Heuristic, PPO, and DQN algorithms (Case 2).

Fig.2 shows the performance comparison of Heuristic, PPO, and DQN algorithms for Case 2. It highlights the adaptability of RL methods to different traffic patterns, showcasing their robustness in various scenarios.

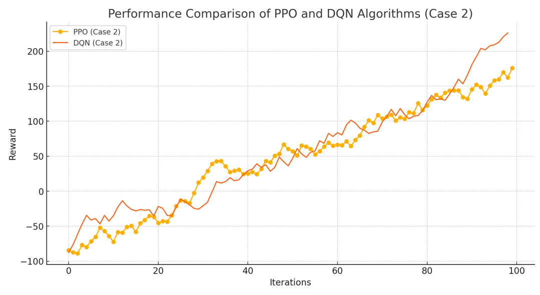

Fig. 3. Performance comparison of PPO and DQN algorithms (Case 2).

Fig.3 compares the performance of PPO and DQN specifically for Case 2, illustrating their convergence patterns. Both algorithms show significant improvement over time, with DQN exhibiting a slightly higher variance in rewards.

Fig.4 provides a general comparison of PPO and DQN, showing their overall performance improvement over time. Both algorithms converge to high reward values, indicating their effectiveness in optimizing network utilization.

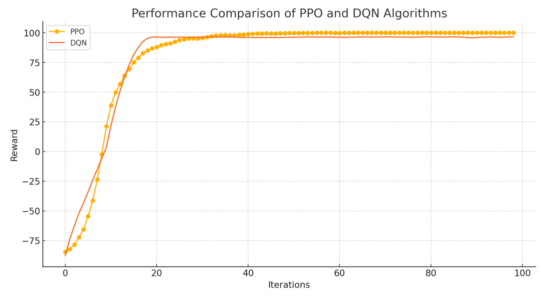

Fig. 4. Performance comparison of PPO and DQN algorithms.

The results show that both PPO and DQN algorithms outperform the heuristic algorithm. This can be attributed to the following reasons:

- **Adaptability**: RL algorithms continuously learn and adapt to the changing network conditions, which allows them to discover and exploit more efficient routing strategies.
- **Exploration and Exploitation**: PPO and DQN balance exploration of new routing paths and exploitation of known successful paths, leading to better overall performance.
- **Learning from Experience**: RL algorithms improve over time as they learn from the rewards and penalties received for successful and blocked requests, respectively. This learning process is not present in the heuristic algorithm, which follows a static strategy.

V. Conclusion

The RL-based routing methods (PPO and DQN) demonstrate significant improvements in network utilization compared to the heuristic algorithm. PPO and DQN effectively learn optimal routing strategies that minimize blocking and maximize utilization. The heuristic algorithm, while simpler, provides a useful benchmark for evaluating the RL-based methods. These results highlight the potential of RL in optimizing complex network environments.

1 Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT Press.

2 Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347.

3 Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

4 Li, Y., & Malik, M. (2017). An overview of machine learning in communications: A tutorial. IEEE Communications Surveys & Tutorials, 19(2), 855-876.