CS258 Final Report: The RSA Problem

Brian Tran and Simar Ghumman
Group 10

I. Methods: RL-based Routing

A. RL Algorithms

Q-Learning: A model-free RL algorithm that learns the optimal action-value function by iteratively updating Q-values based various actions that is taken throughout the steps. The agent will learn from the cumulative reward and make a better decision as iterations progress.

B. State Space

The observation space is comprised of links and requests. The links have a state of all the edges of the graph and we can dictate the space. In addition, there are the source and destination nodes that are initialized in the beginning and then reset every time the reset method are called. Along with that, the holding time is also stored as a variable in the environment.

We can depict a link state as a 10x10 matrix. Once a request goes through and not be blocked, the edges that it takes are filled in with the request's respective holding time. It will be updated at the beginning of every step to be decremented by 1, following Assignment 4's pattern of holding time adjustment

C. Action Space

Three actions are available for the agent to take. The first one will be the blocking action that will only be taken if there are no available paths from the source to the destination in addition to any edge being at max capacity, disabling the agent from establishing a path. The agent is allowed to take one of the following actions.

From the graph visualization, we can see that it is a connected graph, so the first step action will always be guaranteed. However, for the second action, there isn't a guarantee for a second path. However, if the agent takes it, the reward will be doubled to -2, compared to -1. This allows the agent to learn to not try to take unknown paths and punish them more heavily. This will help the non-zero learning rate agent have better results in the long run.

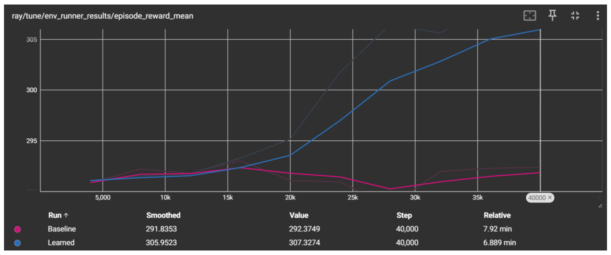
D. Reward Function

The reward function that we decided to use is defined by the following equation:

$$\text{Reward} = \frac{2}{Average\ Path's\ Link\ Utilization}$$

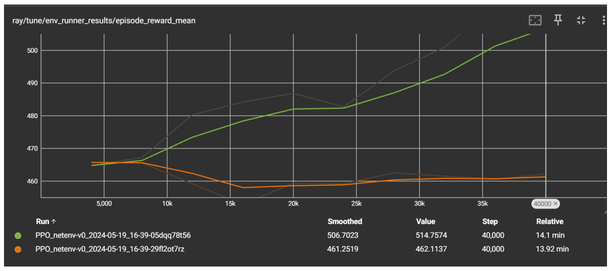
II. Method: Spectrum Allocation

Our heuristic is to provide two possible paths and give a reward based on the reward formula above. The reason we chose this heuristics is because it will provide a path with overall higher capacity in order to avoid potential congestion.



	Run ↑	Smoothed	Value	Step	Relative
●	Baseline	291.8353	292.3749	40,000	7.92 min
●	Learned	305.9523	307.3274	40,000	6.889 min

Fig. 1. Reward vs Episode: Case I



	Run ↑	Smoothed	Value	Step	Relative
●	PPO_netenv-v0_2024-05-19_16-39-05dqq78t56	506.7023	514.7574	40,000	14.1 min
●	PPO_netenv-v0_2024-05-19_16-39-29fl2ot7rz	461.2519	462.1137	40,000	13.92 min

Fig. 2. Reward vs Episode: Case II

If we were to not take this into account and use factors like path length, congestion would easily start. That is where we have our inverted relationship, rewarding higher to small path average.

We chose 2 because we wanted to scale and have a higher reward to accommodate the learning agent. We kept it lightweight but simple. If we wanted to be more advanced, we would taken account of centrality of the nodes. These will give more insight into which nodes will cause congestion and could potentially alter what path the agent will take.

III. Results

A. Learning Curve

Scenario A As shown in Fig. 1, the mean reward for the learning rate agent has a better value compared to the baseline algorithm.

Scenario B As shown in Fig. 2, we can see that the mean reward is also much better for the trained agent compared to the initial non-learning rate agent. We analyzed this to be the case throughout both experiments.

B. Utilization (The Objective)

Here we decided to take the best-fit line and graph it because having 40,000 dots will not be readable. Keep in mind that there are higher values at times and could be skewed by the data.

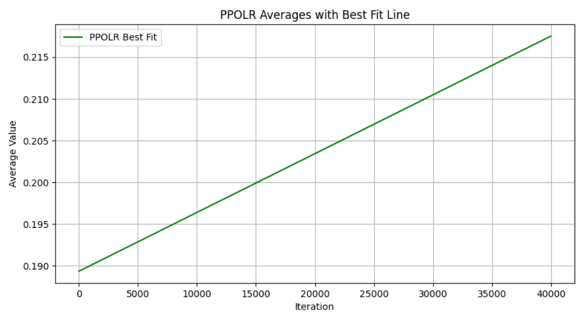


Fig. 3. PPO Learning Rate Best-Fit Link Utilization

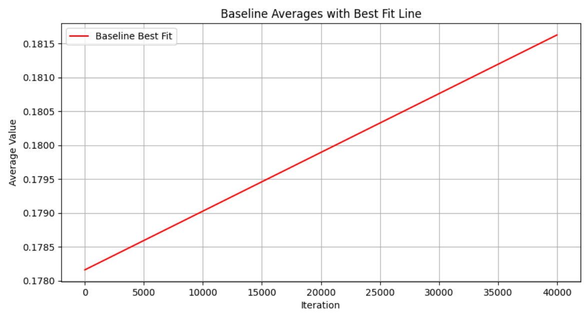


Fig. 4. PPO Baseline Best-Fit Link Utilization

PPO with Learning Rate: It is observed that PPO with learning does much better than the baseline that doesn't reach past 0.2. We can see the improvement that it reaches pass 0.22 and is better at choosing the path for the request.

PPO Baseline: We notice how the baseline isn't the best as it is under 0.19. We can analyze how it doesn't learn and would take potential random actions like blocking when it is not needed and taking routes that are not valid and/or full.

DQN: We analyze how the DQN does the best on average compared to the other two. The average link utilization goes up to 0.24. We believe that this is potentially one of the best we could do with the lightweight heuristics. We could definitely improve on link utilization in the future

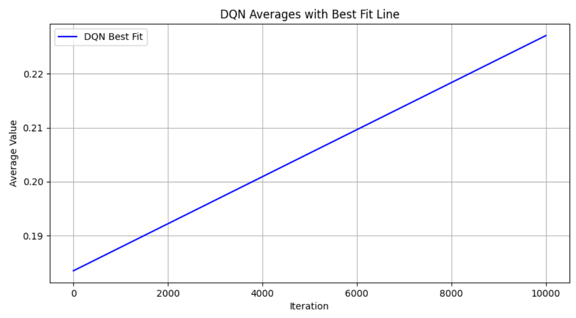
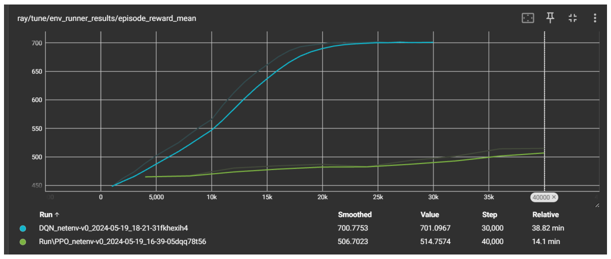


Fig. 5. DQN Best-Fit Link Utilization



Run ↑	Smoothed	Value	Step	Relative
DQN_netenv-v0_2024-05-19_18-21-31fkhexih4	700.7753	701.0967	30,000	38.82 min
Run\PPO_netenv-v0_2024-05-19_16-39-05dqq78t56	506.7023	514.7574	40,000	14.1 min

Fig. 6. DQN vs PPO Reward

C. *Comparison*

PPO vs DQN: In the experiments we ran earlier, we used the **PPO (Proximal Policy Optimization)** Configuration for our model. PPO is a method that is mainly applied to continuous action space, with compatibility with discrete action space.

On the other hand, **DQN (Deep-Q Network)** is a model that is designed specifically for discrete action spaces.

The differences between the two are that DQN learns from a value-based method while PPI uses a policy that maps states to actions and PPO are more stable with its clipped objective function while a replay mechanism backs DQN's stability.

As we can analyze in Fig. 6, DQN is a much better-performing model as the reward average increases higher and quicker compared to the PPO method. One reason for this could be that DQN is better fitted for this discrete space since the actions are discrete.

IV. Conclusion

We can analyze that DQN is a better Rllib implementation than PPO and that with adjusting heuristics and different learning rate, we can accomplish a higher link utilization model and increasing reward vs. episode.