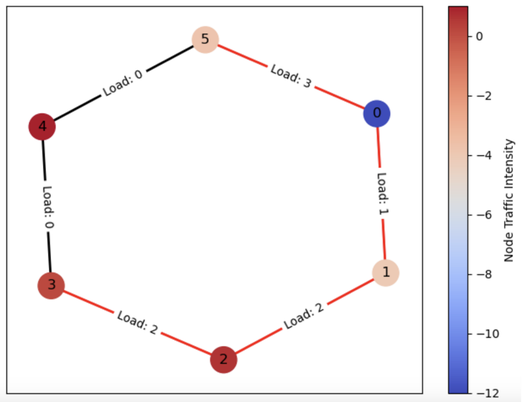
# Reinforcement Learning for Real-Time Network Traffic Path Optimization

Krishnanand, Anip Kumar Paul and Saumya Pandey (CSE 546 Reinforcement Learning by Alina Vereshchaka)

## Introduction

This project improves the real time traffic networking and optimizing the routing paths using the ring topology. We have designed a custom ring Network environment to simulate out the traffic network, where the agent will choose the routing path by optimizing the shortest distance.




## Environment Design

### Custom NetworkEnvironment Class

- Custom Network Simulation: We designed the customized environment using NetworkX and OpenAI Gym so that this environment simulates realistic, dynamic network conditions.
- Flexible Learning Space: This environment allows the RL agent to learn and adapt to real-time fluctuations in traffic, enhancing network routing efficiency.

### Key Features

- Non-Stationary Complexity: Our network environment is non-stationary and adding complexity by constantly due to changing the traffic loads.
- Ring Topology: We choose ring topology that is challenging and helps the agent learn optimal routing strategies with minimal congestion.
- Detailed State Representation: We provide comprehensive details, such as current node, traffic intensity, adjacent load, and distance to the destination that guiding efficient routing decisions.
- Reward Structure for Optimization: We design the rewards to penalize the latency and congestion, while incentivizing successful packet delivery to the destination.

## RL Algorithm

### Q-Learning

- We use the Q-learning as our baseline model to understand the basic routing dynamics and to get a foundational understanding of network behavior.
- This algorithm fits well initially since the custom environment is discrete, and Q-tables are effective for learning optimal actions in smaller scenarios and evaluates the reward structure based on latency and congestion penalties. The Q-Learning framework:

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal,\cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Take action $A$, observe $R$, $S'$
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until $S$ is terminal

Target · Prediction · Immediate Reward · loss

### Double Deep Q-Network (DDQN)

- As we consider more complex state space with multiple nodes, paths, and dynamic conditions, DDQN is used to approximate Q-values with neural networks, which is beyond the capacity of Q-tables.
- DDQN overcomes this by decoupling action selection and evaluation, which is highly applicable in high-dimensional input features such as traffic intensity, maximum adjacent load, and network topology effectively. DDQN tends to converge more fast and more effective to reach the optimal policies by reducing the overestimation. The Q-Learning framework:

Initialize $Q^A$,$Q^B$,$s$
**repeat**
  Choose $a$, based on $Q^A(s,\cdot)$ and $Q^B(s,\cdot)$, observe $r$, $s'$
  Choose (e.g. random) either UPDATE(A) or UPDATE(B)
  **if** UPDATE(A) **then**
    Define $a^* = \arg\max_a Q^A(s',a)$
    $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)\big(r + \gamma Q^B(s',a^*) - Q^A(s,a)\big)$
  **else if** UPDATE(B) **then**
    Define $b^* = \arg\max_a Q^B(s',a)$
    $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)\big(r + \gamma Q^A(s',b^*) - Q^B(s,a)\big)$
  **end if**
  $s \leftarrow s'$
**until** end

### Actor-Critic (A2C)

- We applied a policy-based (A2C) learning algorithm for real-time routing. A2C is applied for real-time optimization and fine-tuning the network traffic routing strategy by combining the actor and critic to learn optimal routes and evaluate actions.
- The non-stationary nature of the network environment become benefited with the continuous adaptability of A2C that can make it more efficient in learning nuanced behaviors compared to discrete action-value estimations.

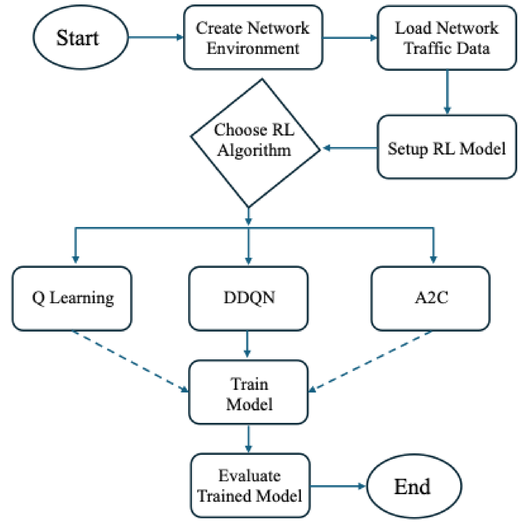
The A2C framework:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s},\mathbf{a},\mathbf{s}',r)$
2. update $\hat{V}^\pi_\phi$ using target $r + \gamma \hat{V}^\pi_\phi(\mathbf{s}')$
3. evaluate $\hat{A}^\pi(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma \hat{V}^\pi_\phi(\mathbf{s}') - \hat{V}^\pi_\phi(\mathbf{s})$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s},\mathbf{a})$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

## Data Set

We choose a Kaggle data set named Synthetic Network Traffic Dataset (Ref: 2). This dataset is designed for network analysis scenarios like network intrusion detection systems, traffic load analysis, dynamic path optimization, and routing strategy testing.

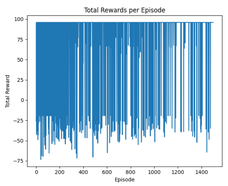
## Model Flow Chart

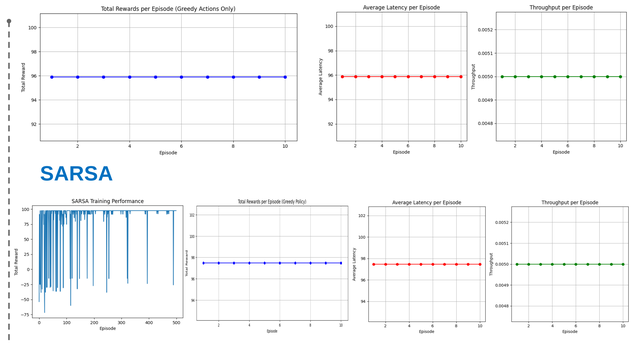



## Results

### Q-Learning

```
Episode 1/1500, Total Reward: -25.78, Epsilon: 0.999
Episode 2/1500, Total Reward: -19.31, Epsilon: 0.998
Episode 3/1500, Total Reward: 95.89, Epsilon: 0.997
Episode 4/1500, Total Reward: -19.31, Epsilon: 0.996
Episode 5/1500, Total Reward: -42.73, Epsilon: 0.995
Episode 6/1500, Total Reward: -19.31, Epsilon: 0.994
Episode 7/1500, Total Reward: 95.89, Epsilon: 0.993
Episode 8/1500, Total Reward: 76.42, Epsilon: 0.992
Episode 9/1500, Total Reward: 95.89, Epsilon: 0.991
```

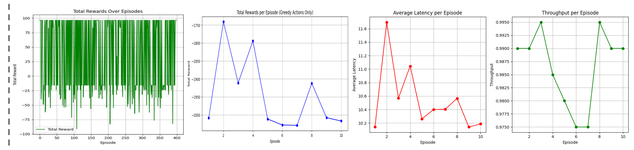


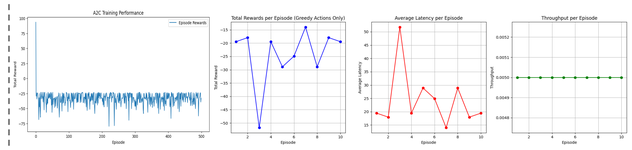

SARSA




DDQN




A2C




## Future Approach

- Apply more optimization on this Traffic Network Environment.
- Integrating more advanced RL algorithms like Proximal Policy Optimization (PPO) or Soft Actor-Critic (SAC).
- Extend the network to more complex network like mesh grid with the multi agent reinforcement learning.
- Explore more complex reward structures with energy consumption, packet loss, quality of service.
- Apply adaptive network changes, like nodes joining or leaving the network.

## References

1. Wu, Q., Wu, J., Shen, J., Yong, B. and Zhou, Q., 2020. An edge based multi-agent auto communication method for traffic light control. Sensors, 20(15), p.4291.
2. Kaggle Synthetic Network Traffic Dataset (https://www.kaggle.com/datasets/sarbojitdas/synthetic-network-traffic)
3. RL class lectures by Dr. Alina Vereshchaka
4. Zheng, Y., Luo, J., Gao, H., Zhou, Y. and Li, K., 2025. Pri-DDQN: learning adaptive traffic signal control strategy through a hybrid agent. Complex & Intelligent Systems, 11(1), p.47.
5. Konda, V. and Tsitsiklis, J., 1999. Actor-critic algorithms. Advances in neural information processing systems, 12.
6. Van Hasselt, H., Guez, A. and Silver, D., 2016, March. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 30, No.1)

University at Buffalo The State University of New York
Department of Computer Science
cse.buffalo.edu