Policy Gradient Reinforcement Algorithms in Network related Use cases
(CS391 - Design Project 2024)

Vedulla Sai Vardhan 202251156@iiitvadodara.ac.in
Vankayala Koutilya 202251152@iiitvadodara.ac.in
Yallapu Vignesh 202251161@iiitvadodara.ac.in
Utkarsh Rana 202251148@iiitvadodara.ac.in

***Abstract*—For networks to function well, reduce latency, and balance traffic loads, efficient routing is essential. The dynamic nature of contemporary networks frequently prevents traditional static and heuristic-based routing techniques from adapting, resulting in less-than-ideal performance in situations with varying traffic. In these kinds of settings, reinforcement learning (RL), and in particular Q-learning, provides a strong foundation for adaptive decision-making.**

Applying Q-learning to network routing decision optimization is the main goal of this research. The routing problem is modeled as a Markov Decision Process (MDP), and the algorithm uses real-time feedback in the form of incentives based on network performance measures like latency and congestion to repeatedly learn optimum policies by interacting with the environment. By using the Q-learning technique, routers may update their routing tables dynamically, reducing path selection expenses while maintaining efficiency and dependability.

***Index Terms*—REINFORCE, MDP, PPO, Reinforcement Learning, Policy Gradient Algorithms, Q-Learning, Route Optimization**

I. Introduction

Efficient management and Optimisation are key for maintaining high performance, scalability and adaptability to changing Networking footprints in modern networking. Static rule-based techniques or algorithm- driven optimization solutions adopt traditional approaches to network optimization, but they cannot adapt to the versatility and uncertainty of the network ecosystem. The introduction of reinforcement learning (RL), especially policy gradient algorithms, has brought a new approach on solving networking problems.

Reinforcement learning policy gradient methods such as REINFORCE, Proximal Policy Optimization (PPO), and Trust Region Policy Optimization (TRPO) operate by directly optimizing a policy by teaching the agent to take actions which maximize cumulative reward. Unlike the value-based methods such as Q-learning, where the main goal is to compute action values return and use them to choose a proper direction when further improving the policy, these methods are distributional based at their core due to constructing probabilities over actions making them highly effective in continuous action spaces and high dimensional problems. For networking applications, these algorithms could leave space for solving problems like dynamic routing, resources allocation and traffic management as the conditions change.

Route optimization in IP networks is a strong application for policy gradient algorithms, as network factors including connection failures, and varying traffic patterns need real-time flexibility. Through the balance of exploration and exploitation, policy gradient approaches allow routers to discover optimum pathways, which improves load distribution, performance, and latency. Likewise, in wireless networks, similar techniques may control the distribution of resources, including spectrum and power, to improve signal quality and reduce interference.

The goal of this research is to solve network optimization problems, namely optimised routing, by utilizing policy gradient reinforcement learning. In order to show that these approaches may perform better than conventional solutions in dynamic, large-scale networks, it is intended to investigate their efficiency and flexibility. In order to contribute to the creation of intelligent, self-optimizing networks, this study attempts to close the gap between theoretical developments in reinforcement learning and real-world applications in network settings.

II. Concepts Explored

A. *Reinforcement Learning Overview*

A machine learning paradigm known as reinforcement learning (RL) teaches an agent to make successive decisions by interacting with its surroundings. In contrast to supervised learning, which uses labeled datasets, reinforcement learning makes use of trial-and-error techniques. RL is especially well-suited to handling dynamic and complicated issues where conventional approaches are inadequate since the agent's objective is to maximize cumulative rewards over time.
Core Concepts in Reinforcement Learning:

- **State**: The environment's representation at a given time, which the agent perceives.
- **Action**: The decision the agent makes in a particular state to influence the environment.
- **Reward**: Feedback from the environment evaluating the agent's action, either positive or negative, that helps guide future decisions.
- **Policy**: A strategy that defines the agent's behavior by mapping states to actions to maximize rewards. Policies can be deterministic or stochastic.

The state-value function and action-value function (Q-function) are two important metrics used in reinforcement learning to assess the predicted rewards from specific states or state-action pairings. The optimization of policies is driven by these metrics. The unique benefit of RL is its capacity to manage sequential decision-making tasks in unpredictable and changing environments.

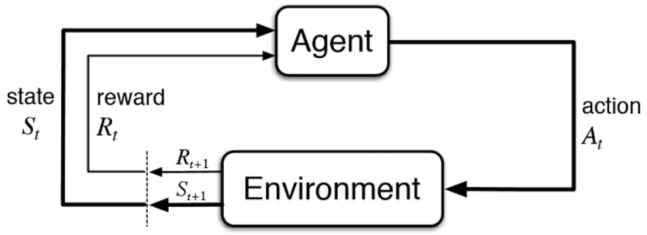


Fig. 1. Reinforcement Learning

B. *Policy Gradient Methods*

Policy gradient methods, a subclass of reinforcement learning, optimise the agent's policy directly.Policy gradient approaches allow agents to more efficiently enhance their decision-making processes by modifying the probabilities of actions using gradients of expected rewards.
Key advantages of policy gradient methods include:

- Handling dynamic and uncertain environments efficiently.
- Supporting stochastic policies, which are essential for networks where conditions frequently change.
- Offering scalability for high-dimensional tasks, such as optimizing resource allocation or managing traffic in real-time.

C. *Applications in Network Routing*

In networking, RL offers a flexible and dynamic framework for the best routing choices, which is crucial in settings where topology, load, and latency are all subject to change. These intricacies are frequently too much for conventional static or rule-based routing techniques to manage.
RL-based routing enables:

- **Latency Reduction**: By choosing paths with minimal delay.
- **Dynamic Path Management**: Adapting to real-time changes in network conditions.
- **Optimal Resource Utilization**: Balancing load across multiple paths for efficient network performance.

D. *Key Strategies for Path Optimization*

Optimal path routing involves finding the most efficient paths for data packets based on parameters such as:

- Distance (or Cost): Measures the total cost of a route, including factors like bandwidth consumption or link quality. Shorter paths do not always equate to lower costs due to varying link conditions.
- Latency (or Delay): The time it takes for packets to travel from the source to the destination. Latency can fluctuate over time, making adaptive routing crucial.

Numerous strategies exist for determining the shortest paths, such as:

- Static Algorithms: Traditional methods like Dijkstra's algorithm are deterministic and assume fixed network conditions. However, they fail in dynamic environments or graphs with negative weights.
- Dynamic Algorithms: These adapt to real-time changes in network topology and conditions.
- Reinforcement Learning: A model-free approach that uses experience to optimize routing decisions. RL-based methods outperform traditional algorithms by adapting to non-stationary environments.

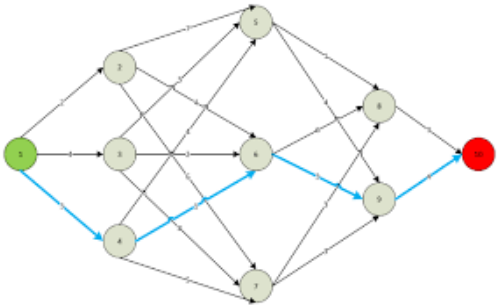


Fig. 2. Route Optimisation

E. *Markov Decision Processes (MDPs)*

RL applies the concept of Markov Decision Processes (MDPs) to solve the routing problem.
An MDP consists of:

- States: Represent different network configurations.

- Actions: Correspond to routing decisions.
- Rewards: Signal the quality of the decision based on metrics like reduced latency or lower cost.

The agent optimizes routing policies by iteratively evaluating state-action pairings using reinforcement learning. The state-value function and the action-value function (Q-function) are important equations.

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_t = s\right] \tag{1}$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_t = s, A_t = a\right] \tag{2}$$

F. *Q-Learning and the Q-Function*

Q-learning is an off-policy reinforcement learning algorithm used to learn the value of an action in a particular state. The value is represented by the Q-function, Q(s,a), which estimates the expected future reward of taking action a in state s and following the optimal policy thereafter. The Q-function is updated iteratively using the Bellman optimality equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{3}$$

where:

- α is the learning rate.
- r is the reward received after taking action a in state s.
- γ is the discount factor that balances immediate and future rewards.
- s' is the resulting state after taking action a.

The Q-learning algorithm converges to the optimal Q-function, $Q^*(s,a)$, which provides the maximum expected reward for each state-action pair

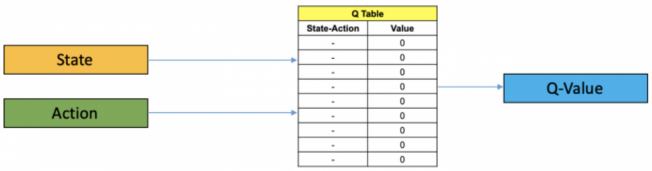


Fig. 3. Q-Learning Visualisation

G. *Modified Q-learning for optimal path routing*

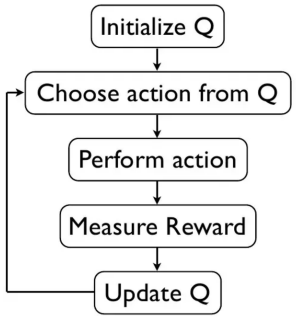
The value of a state-action pair is kept in a Q-table by the Q-learning method. The Q-learning gathers the values of specific state-action pairs in a table during the state space exploration, which will help in determining the best course of action.

Since the Q-learning algorithm needs to be adjusted to meet minimization constraints, the optimal path routing problem can be viewed as a minimization problem. As a result, the previously stated Q-learning equation can be modified as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \min_{a'} Q(s',a') - Q(s,a)\right] \tag{4}$$

The updated Q-learning method can alternatively be changed as follows since the discount factor γ is a case-by-case requirement and is not a required component

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \min_{a'} Q(s',a') - Q(s,a)\right] \tag{5}$$



H. *Q-learning algorithm for optimal path routing*

Here is the modified Q-learning algorithm for finding the optimal path

Algorithm 1 Q-Learning Algorithm for optimal path routing

0: Initialize $Q(s,a)$ arbitrarily for all states $s \in S$ and actions $a \in A(s)$
0: Set $Q(s,a) \leftarrow 0$ for all s, a if no prior knowledge exists
for *each episode* **do**
0:
Initialize starting state s **while** *the episode is not terminated* **do**
0:
Choose an action a for state s using a policy (e.g., ϵ-greedy)
0: Take the action a, observe reward r, and next state s'
0: Update $Q(s,a)$ using:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \min_{a'} Q(s',a') - Q(s,a)\right]$$

0: Set $s \leftarrow s'$ until s is terminal
0:
0: =0

METHODOLOGY

I. *Problem Formulation*

The network routing problem is modeled as a *Markov Decision Process (MDP)*, which is defined by:

- **States** (S): Each state represents the current node in the network.
- **Actions** (A): Actions correspond to selecting the next hop for a packet at a given node.
- **Reward** (R): The reward is based on the cost of traversing an edge, such as latency, congestion, or packet loss. A lower cost leads to a higher reward.

- **Transition** (T): The next state depends on the selected action, representing the movement of a packet to the next node.

The objective is to learn an *optimal policy* (π^*) that minimizes the cumulative cost while ensuring efficient routing.

J. Q-Learning Algorithm Implementation

The Q-learning algorithm is used to find the optimal policy through the iterative update of Q-values:

1) **Initialization:**
   - The Q-value table $Q(s, a)$ is initialized arbitrarily for all state-action pairs.
   - Learning rate (α), discount factor (γ), and exploration rate (ϵ) are set.
2) **Action Selection:** Actions are selected using an ϵ-greedy policy to balance exploration (choosing random actions) and exploitation (choosing actions with the highest Q-value).
3) **Reward Feedback:** For each action taken, the agent receives a reward based on the cost of the edge traversed.
4) **Q-Value Update:** The Q-value for the state-action pair is updated using the previous given formula.
5) **Policy Extraction:** After convergence, the policy selects the action with the highest Q-value for each state.

K. Experimental Setup

- **Structure:** A directed seuence with nodes representing routers and edges representing links with associated costs.
- **Start and Goal Nodes:** The source and destination for routing packets were predefined.
- **Performance Metrics:**
  - **Path Cost:** The total cost of the route selected.
  - **Convergence:** The number of episodes required to stabilize the Q-values.
  - **Optimality:** The ability to find the shortest or most efficient path.

L. Results and Comparisions

- The final policy was evaluated to determine its performance in finding optimal routes under different network conditions.
- Comparisons were made with traditional routing algorithms (e.g., Dijkstra's algorithm) to highlight the adaptability of Q-learning to dynamic conditions such as fluctuating edge costs or link failures.

This methodology emphasizes adaptive learning and performance gains above conventional approaches, outlining a systematic method to using Q-learning for network route optimization.

CONCLUSION

This study shows how reinforcement learning—more especially, the Q-learning algorithm—can be used to the network route optimization issue. When it comes to dynamic network situations like shifting traffic loads, connection outages, or topological changes, traditional static routing techniques frequently fall short. Q-learning offers a strong framework for learning and optimizing routing rules based on real-time feedback from the environment by simulating the routing issue as a Markov Decision Process (MDP).

The outcomes demonstrate how well Q-learning works to increase routing efficiency. The system dynamically adjusts to changing network circumstances through repeated exploration and exploitation, producing notable gains in metrics like balanced traffic distribution, decreased latency, and lower route costs. Q-learning actively learns from its surroundings and modifies its decisions to guarantee optimal performance even in non-stationary network settings, in contrast to conventional algorithms like Dijkstra's that depend on static inputs.

In summary, our study bridges the gap between theoretical reinforcement learning models and real-world networking applications by laying the groundwork for using Q-learning to route optimization. The results show that reinforcement learning can revolutionize the way contemporary networks function by enabling intelligent, adaptable, and self-optimizing routing algorithms when customized to the unique requirements of network environments.

ACKNOWLEDGEMENT

We would like to sincerely thank our mentor, Dr. Bhupendra Kumar, for his essential assistance, encouragement, and direction during this project. His knowledge and perceptions have greatly influenced our understanding of reinforcement learning and how it applies to network route optimization. His mentoring and helpful criticism have significantly improved the quality of our work and motivated us to expand our expertise.

Lastly, we would like to express our sincere gratitude to our friends and the academic community for creating a friendly and cooperative environment that has been crucial to the accomplishment of this project.

REFERENCES

[1] Richard S. Sutton and Andrew G. Barto. 2018. Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA.
[2] Jonathan Hui's Medium Article – RL Policy Gradients Explained
[3] Scholarpedia – Policy Gradient Methods
[4] Watkins, C.J .C.H, & Dayan, P. (1992). Q-Iearning. Machine Learning.
[5] Rajasekhar Nannapaneni - Optimal Path Routing Using Reinforcement Learning