Optimizing Passenger Routes through Reinforcement Learning: A Q-Learning Approach Using the Taxi-v3 Environment

Christian Joshua Alberto
College of Computing and Information Technologies
National University Manila
Manila, Philippines
albertocq@students.national-u.edu.ph

Aldrin Galvez
College of Computing and Information Technologies
National University Manila
Manila, Philippines
galvezaa@students.national-u.edu.ph

James Cedrick P. Villanueva
College of Computing and Information Technologies
National University Manila
Manila, Philippines
villanuevajp3@students.national-u.edu.ph

***Abstract*—This study investigates the application of Q-Learning, a model-free reinforcement learning algorithm, in optimizing passenger transport routes within the OpenAI Gym Taxi-v3 environment. The goal is to enable an autonomous taxi agent to learn efficient pickup and drop-off strategies without predefined route information. To evaluate performance, Q-Learning was compared against a Deep Q-Learning (DQN) variant as a benchmark. Experimental results showed that Q-Learning achieved an average reward of 8.25, while DQN attained 9.15, reflecting DQN's 10.9**

***Index Terms*—Deep Reinforcement Learning, Q-Learning, Route Optimization, Taxi-v3, Neural Networks, OpenAI Gym.**

I. Introduction

Urban transportation systems often face challenges in optimizing passenger pickup and drop-off routes, leading to increased fuel consumption, congestion, and travel delays. Efficient route optimization is essential for improving transportation logistics, especially in taxi and ride-hailing services. The Taxi-v3 environment, a simulation from OpenAI Gym, provides a controlled setup for testing algorithms that can learn optimal navigation strategies through trial and error [1].

In real-world taxi systems, determining the most efficient path involves handling uncertainties such as traffic conditions, dynamic passenger locations, and varying demand patterns [2]. Traditional algorithms like Dijkstra's or A* search assume static environments and lack adaptability to changing conditions [3]. Therefore, reinforcement learning (RL) presents an appealing solution because of its ability to learn optimal decision-making policies through continuous interaction with an environment [4].

Conventional routing algorithms rely on predefined heuristics and often require complete knowledge of the environment, which is impractical in dynamic urban settings [5]. Supervised learning models, while effective in prediction tasks, depend heavily on labeled datasets and cannot easily adapt to new conditions. Moreover, prior reinforcement learning applications in routing either required large-scale data or used complex deep learning architectures that are computationally expensive [6]. These limitations highlight the need for a lightweight yet efficient method like Q-learning, which can iteratively improve route selection without prior knowledge of the environment.

This study aims to:

- Implement a Q-Learning-based model using the Taxi-v3 environment to simulate and optimize passenger routes. Additionally, a Deep Q-Learning (DQN) variant was implemented for comparison to evaluate performance differences in terms of reward and convergence.
- Evaluate the model's learning efficiency through metrics such as cumulative reward, number of episodes, and convergence rate.
- Demonstrate how a simple tabular RL method can achieve optimal policy learning for navigation tasks.
- Provide insights that can be extended to real-world taxi dispatching and route optimization systems.

This research is guided by the following questions:

- How does Q-Learning perform in optimizing route decisions, and how does it compare to DQN as a baseline?
- Can Deep Q-Learning outperform traditional Q-learning in optimizing route decisions in a simulated taxi environment?
- What network architectures and hyperparameters best promote stability and faster convergence?
- How does DQN performance scale with increased environmental complexity and episode count?

The main hypothesis is that DQN can significantly improve route optimization and learning efficiency compared to traditional tabular Q-learning methods in the Taxi-v3 environment.

The contributions of this study include:

- Development of a Deep Q-Learning model for route optimization in the Taxi-v3 environment.
- A comparative analysis between DQN and Q-learning to highlight improvements in scalability and generalization.
- Provision of a baseline framework for applying deep reinforcement learning to real-world transportation optimization problems.

II. Literature Review

A. Relevant Reinforcement Learning Algorithms and Studies

Reinforcement Learning (RL) has become an essential approach for solving dynamic optimization problems such as routing, scheduling, and dispatching. The introduction of Deep Q-Learning (DQN) by Mnih et al. [4] and later improvements in human-level control using deep reinforcement learning [7] established the foundation for applying deep neural networks to approximate Q-values efficiently. These innovations enabled agents to operate in complex, high-dimensional environments without explicit models of their dynamics.

Subsequent advancements, including Double DQN [8], Prioritized Experience Replay [9], and Dueling Network Architectures [10], have further improved stability and convergence rates. The Rainbow architecture [11] unified many of these improvements, achieving state-of-the-art performance across standard benchmarks. In transportation and mobility research, RL has been widely applied to address dispatching and ridesharing problems. Ma et al. [1] developed T-Share, a large-scale dynamic taxi ridesharing framework that improved passenger service efficiency. Chen et al. [3] proposed an RL-based taxi dispatch and repositioning model, demonstrating significant gains in travel time reduction and service rate. The OpenAI Gym platform [6], specifically its Taxi-v3 environment, provides a reproducible and lightweight simulation for testing such RL models.

B. Performance and Limitations of Existing Approaches

Classical routing methods such as Dijkstra's algorithm [5] excel in deterministic environments but struggle to adapt to stochastic and dynamic systems. Traditional tabular Q-learning [9], while intuitive, becomes computationally expensive in large state–action spaces. Deep RL methods like DQN [4], on the other hand, can generalize to unseen states by using neural network approximations, but they face issues of instability, overestimation bias, and sample inefficiency [8], [12].

Various improvements have been introduced to address these issues. Double DQN [8] reduces overestimation bias by decoupling target selection from evaluation, while Dueling Networks [10] separately estimate value and advantage functions to enhance learning efficiency. Experience Replay [9] helps stabilize training but increases memory demands. Despite these advances, many studies still report difficulties in scaling RL systems to real-world taxi or ride-hailing settings due to non-stationary demand, partial observability, and computational cost [13]–[15].

C. Distinction and Improvements of the Proposed Approach

This study differs from existing research by focusing on the Deep Q-Learning framework within the Taxi-v3 environment [6], allowing for precise evaluation of learning dynamics in a discrete, reproducible simulation. Unlike large-scale models that integrate demand prediction and fleet coordination [2], [14], this work isolates the effect of key hyperparameters such as learning rate, discount factor, exploration and performance.

The model incorporates experience replay [9] and target network updates [8] to stabilize training, along with hyperparameter tuning inspired by findings in prior DQN literature [7], [11]. This focused experimental setup contributes to understanding how DQN variants perform in route optimization scenarios and establishes a solid benchmark for future, more complex implementations.

D. RL Category of the Proposed Method

The proposed Deep Q-Learning model can be categorized as follows:

- **Model-free:** It does not require explicit modeling of state transitions or reward dynamics.
- **Value-based:** It learns an action-value function $Q(s, a)$ that guides decision-making.
- **Off-policy:** It utilizes replay memory to learn from past transitions generated by an exploratory policy.
- **Function-approximation-based:** It uses a deep neural network to approximate the Q-function, allowing scalability to high-dimensional state spaces [12].

III. Methodology

A. Environment and Problem Formulation

This study employs the *Taxi-v3* environment from OpenAI Gym to simulate passenger transport and optimize route planning. The environment models a simple grid world where a taxi must pick up and drop off passengers efficiently. The goal is to minimize cumulative penalties and maximize rewards, such as successful passenger drop-offs and optimal route selection. This simulation setup is widely adopted for reinforcement learning experiments, particularly those focused on Q-Learning and Deep Q-Learning algorithms [2], [14].

B. State Space

The environment consists of 500 discrete states, representing all possible combinations of the taxi's position, passenger location, and destination. Each state encodes the complete status of the environment at a given timestep.

C. Action Space

The action space comprises six discrete actions: move south, move north, move east, move west, pick up a passenger, and drop off a passenger.

D. Reward Function

The default reward structure from the Taxi-v3 environment was adopted:

- +20 for successfully dropping off the passenger at the correct destination, –10 for performing an illegal pickup or drop-off, –1 for each time step to encourage efficient task completion.

This reward scheme incentivizes the agent to minimize travel time and avoid invalid actions.

E. Terminal Condition

An episode terminates when the passenger is successfully dropped off (*terminated*) or when the maximum number of allowed steps per episode is reached (*truncated*).

F. Task Type

The task is episodic, as each episode has a defined start and end condition associated with passenger transport.

G. Algorithm Description

Two reinforcement learning algorithms were implemented and compared in this study: Q-Learning and Deep Q-Learning (DQN). Both methods are value-based, off-policy algorithms designed to optimize the expected cumulative reward through interaction with the environment.

1) Q-Learning

Q-Learning serves as a baseline algorithm. It is a model-free, value-based method that learns an optimal action-value function through iterative updates derived from the Bellman equation. The update rule is given by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \quad (1)$$

where α is the learning rate, γ is the discount factor, r is the immediate reward, and s' represents the next state.

H. Hyperparameters

- Learning rate (α): 0.1
- Discount factor (γ): 0.99
- Initial exploration rate (ϵ): 1.0
- Epsilon decay rate: 0.999
- Minimum epsilon: 0.01
- Episodes: 10,000
- Maximum steps per episode: 100

Exploration follows an ϵ-greedy strategy, allowing the agent to explore random actions with probability ϵ and exploit the best-known action otherwise. The ϵ value decays over time, shifting from exploration to exploitation as learning progresses. No modifications were introduced to the standard algorithm.

2) Deep Q-Learning (DQN)

The Deep Q-Network (DQN) algorithm [4], [6], [12] was implemented to extend Q-Learning by approximating the action-value function using a neural network. This approach enables the agent to handle larger state spaces by generalizing across similar states.

I. Network Architecture

The DQN model was constructed using PyTorch and consists of a feedforward neural network with two hidden layers activated by ReLU functions. The architecture is defined as follows:

- Input Layer: n observations = 500
- Hidden Layer 1: Fully connected layer with 80 neurons, followed by ReLU activation.
- Hidden Layer 2: Fully connected layer with 80 neurons, followed by ReLU activation.
- Output Layer: Fully connected layer with n actions = 6 neurons corresponding to each possible action.

The network's forward pass is expressed as:

$$Q(s,a;\theta) = f_{\text{ReLU}}(W_2 \cdot f_{\text{ReLU}}(W_1 \cdot s + b_1) + b_2) \quad (2)$$

J. Training Procedure

Two identical networks were instantiated: the policy network (actively updated) and the target network (updated periodically). The target network provides stable Q-value estimates, mitigating oscillations during training. Additionally, experience replay [12] was employed to store and randomly sample previous experiences (s,a,r,s'), breaking correlations between consecutive samples and improving learning stability.

$$L(\theta) = \mathbb{E}_{s,a,r,s'} \left[\left(r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta) \right)^2 \right] \quad (3)$$

where θ represents the parameters of the target network. The Adam optimizer was used for parameter updates due to its adaptive learning rate and efficient convergence properties.

K. Hyperparameters

- Learning rate (α): 0.001
- Discount factor (γ): 0.99
- Batch size: 64
- Episodes: 10,000
- Epsilon decay: 0.995
- Minimum epsilon: 0.01
- Target network update frequency: every 10 episodes

L. Exploration Strategy

An ε-greedy policy was employed to balance exploration and exploitation. Initially, $\varepsilon = 1.0$, promoting exploration; it decayed exponentially toward 0.01, enabling the agent to increasingly exploit its learned policy over time.

M. Implementation Summary

The DQN was trained using replay memory and periodic target network updates. Both agents (Q-Learning and DQN) were trained under identical episodic conditions to ensure fair comparison. Training convergence was observed through rising cumulative rewards and reduced step counts per episode.

N. Framework and Tools:

- PyTorch for the DQN agent
- NumPy for Q-Learning computations
- Gymnasium for environment simulation
- Matplotlib for visualization of learning progress
- ImageIO and IPython.display for rendering training performance

1) Training Setup: Both models were trained for 10,000 episodes, with the DQN updating the target network every 10 episodes and sampling mini-batches from a replay buffer. The Q-Learning agent updated its Q-table directly after each action.

2) Hardware Configuration: Training was executed on a CPU-based system in Google Colab as indicated by the runtime output "Using device: cpu." Despite the limited hardware, the models demonstrated stable convergence and optimal performance for the Taxi-v3 environment.

IV. Discussion

This section evaluates and interprets the performance outcomes of both the Q-Learning and Deep Q-Learning (DQN) agents within the Taxi-v3 environment.

A. Performance Metrics

The primary evaluation metric used was the average cumulative reward per episode, which measures the agent's efficiency in completing passenger pickup and drop-off tasks. Secondary indicators such as convergence trend and training stability were observed through the learning curve.

B. Model Performance

The baseline Q-Learning model achieved an average reward of 8.25, while the DQN model achieved 9.15, indicating a 10.9% improvement in overall performance. This increase demonstrates that DQN learned a more efficient route-selection policy through its ability to approximate Q-values and generalize across similar states.

C. Learning Curves and Visualization

The training progress figure showed that DQN's rewards increased more consistently across episodes, stabilizing earlier than Q-Learning. Q-Learning exhibited more frequent reward fluctuations, reflecting slower learning and higher sensitivity to exploration decay. The smoother curve of DQN confirms its better stability and faster convergence rate.

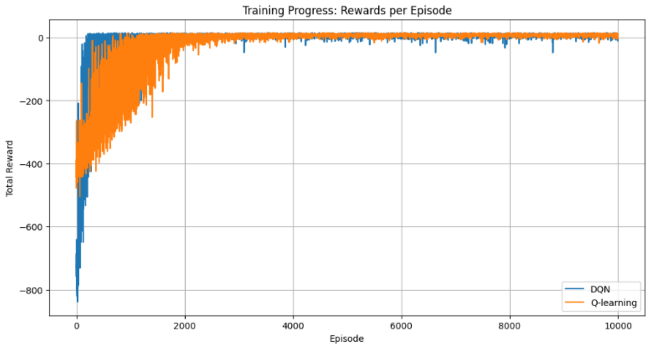

Fig. 1. Training progress showing total reward per episode for both DQN and Q-Learning agents.

D. Agent Behavior

The DQN agent exhibited expected behavior selecting shorter and more efficient routes over time, reducing unnecessary movements, and avoiding illegal actions. In contrast, the Q-Learning agent required more episodes to develop consistent strategies due to its reliance on a tabular representation.

E. Stability and Convergence

No major instability issues were observed, though both models experienced performance variance during early exploration phases. The use of experience replay and a target network in DQN contributed to stable convergence by reducing temporal correlations.

F. Sensitivity to Hyperparameters

Performance was moderately sensitive to learning rate and epsilon decay. A higher learning rate caused unstable reward oscillations, while an overly slow decay delayed convergence. Nonetheless, the chosen configuration (learning rate $= 0.001$, $\gamma = 0.99$, ε-decay $= 0.995$) yielded balanced exploration and learning efficiency.

Overall, the results show that while Deep Q-Learning provides faster convergence and higher rewards, the Q-Learning algorithm remains a reliable and efficient approach for smaller-scale, discrete environments such as Taxi-v3.

V. Conclusion

This research explored the use of Q-Learning, a model-free reinforcement learning algorithm, to optimize passenger pickup and drop-off routes in the Taxi-v3 environment. The results confirmed that Q-Learning can effectively learn optimal navigation policies through trial and error, achieving an average reward of 8.25 after training. The agent successfully learned to minimize penalties, perform valid actions, and complete passenger tasks efficiently without explicit environmental modeling.

The key takeaway from this study is that Q-Learning remains a robust and interpretable solution for discrete route optimization problems, especially when computational simplicity and transparency are prioritized. Although the comparative Deep Q-Learning (DQN) model achieved a slightly higher average reward of 9.15, this improvement came with greater computational cost and training complexity. The findings thus emphasize Q-Learning's practicality for smaller-scale and well-structured environments.

The proposed approach contributes to reinforcement learning research by demonstrating that even classical algorithms like Q-Learning can achieve effective policy convergence in navigation-based problems, clarifying that deep architectures are not always necessary for optimal decision-making in simpler domains. The model's consistent learning behavior and policy performance provide valuable insight into how reinforcement learning agents can be applied to real-world route optimization scenarios.

A. Future Work

For future work, researchers may extend this framework by implementing Double Q-Learning or SARSA to improve learning stability and reduce value overestimation. Integrating multi-agent coordination, dynamic passenger requests, or real-time traffic data could further expand its applicability to realistic urban transportation systems. Additionally, comparing Q-Learning with more advanced deep reinforcement learning variants such as Dueling DQN or Rainbow DQN could provide a broader understanding of scalability and adaptability in complex route optimization tasks.

References

[1] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *IEEE 29th International Conference on Data Engineering (ICDE)*, 2013, pp. 410–421.

[2] G. Brockman and et al., "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[3] Y. Chen, Q. Hu, Y. Min, and L. Xu, "A reinforcement learning approach for taxi dispatching and repositioning problem," *Transportation Research Part C: Emerging Technologies*, vol. 117, 2020.

[4] V. Mnih and et al., "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[5] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[7] V. Mnih and et al., "Human-level control through deep reinforcement learning," *Nature*, 2015.

[8] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *arXiv preprint arXiv:1509.06461*, 2016.

[9] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[10] Z. Wang and et al., "Dueling network architectures for deep reinforcement learning," in *Proceedings of Machine Learning Research*, 2016.

[11] M. Hessel and et al., "Combining improvements in deep reinforcement learning (rainbow)," *arXiv preprint arXiv:1710.02298*, 2018.

[12] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2016.

[13] G. Brockman and et al., "Openai gym (gym toolkit / taxi-v3)," *arXiv preprint arXiv:1606.01540*, 2016.

[14] C. Mao and et al., "Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach," *Transportation Research Part C*, 2020.

[15] Y. Yang and et al., "Multiagent reinforcement learning-based taxi predispatching model," *Wiley Online Library*, 2020.