Reinforcement Learning for Adaptive Traffic Signal Control at Simulated Traffic Intersections

Ken K. Hong
khong18@gatech.edu
Georgia Institute of Technology

Abstract

This research investigates reinforcement learning algorithms for managing traffic signals at a simplified four-way intersection. Model-based methods, like Value Iteration and Policy Iteration, show faster convergence, while model-free techniques such as Q-Learning and SARSA offer greater flexibility. How rewards are structured and when the learning process terminates both play important roles in performance; hyperparameters also influence learning stability. The findings highlight the challenges of scaling these methods to real-world traffic systems and point to promising future directions in multi-agent coordination and adaptive control.

1 Introduction

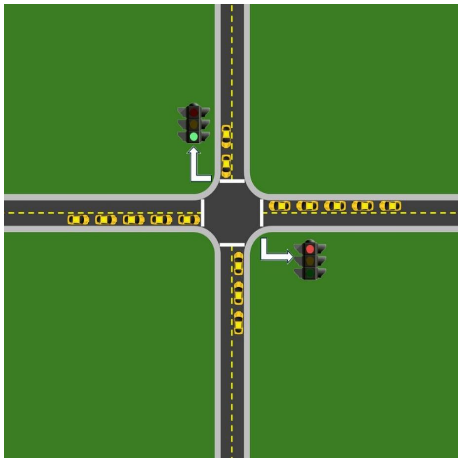
Reinforcement learning (RL) is a powerful approach to sequential decision-making, where agents learn to take actions in dynamic and stochastic environments to maximize long-term rewards. In recent years, RL has seen rapid growth, driven by the broader surge in artificial intelligence research. Notably, the 2024 Turing Award was awarded to Andrew G. Barto and Richard S. Sutton, two pioneers in the field of RL.

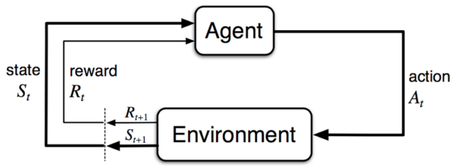
This project explores the application of RL to traffic signal control by optimizing vehicle flow at a simplified intersection using the agent-environment-simulator (EAS) framework. The system is modeled as a Markov Decision Process (MDP), and algorithms such as Value Iteration, Policy Iteration, Q-Learning, and SARSA are implemented to learn effective traffic light control policies. The goal is to reduce congestion by maintaining vehicle queues below critical thresholds, while accounting for stochastic traffic patterns modeled using a Poisson distribution. Through systematic experimentation with various configurations, the project evaluates each algorithm's performance and examines the influence of reward structures, termination conditions, environment dynamics, and hyperparameters on model outcomes and agent behavior.

2 Problem Formulation

2.1 Traffic Intersection Environment

The environment simulates a four-way intersection with two main traffic flows: north-south (NS) and east-west (EW). At any given time, only one direction has a green light, allowing vehicles to pass, while the other direction is stopped with a red light. Vehicle arrivals in each direction follow independent Poisson processes, governed by parameters lambda1(λ_1) and lambda2(λ_2), which represent the expected number of arrivals per timestep. In addition, a constant controls the number of vehicles that leave the intersection when the light is green. The simulator tracks the number of vehicles waiting in each direction, with defined threshold limits for both individual directions and total traffic. The objective of this project is to develop a reinforcement learning agent that decides whether to switch or maintain the traffic light based on the current state of the intersection.


(a) Traffic intersection layout



(b) Reinforcement learning environment

Figure 1: Visualization of the simulation environment

2.2 Markov Decision Process Components

As described by Sutton and Barto (2), this reinforcement learning problem is formulated as a Markov Decision Process (MDP), represented by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- **State space** $(\mathcal{S})$: Represents the current traffic conditions as a tuple, including the number of vehicles queued in the north-south (NS) and east-west (EW) directions, as well as the currently active green light. The state space is finite in this project.

- **Action space** $(\mathcal{A})$: Consists of two possible actions: either maintain the current green light or switch it to the other direction.

- **Transition function** (P): Describes the probabilistic dynamics of the system, influenced by vehicle arrivals modeled as Poisson processes and vehicle departures governed by the traffic light phase. It defines the probability $P(s' \mid s, a)$ of transitioning to a new state s' given the current state s and action a. In this project, the transition function is assumed to be stationary, satisfying the Markov property:

$$P(s_i = s' \mid s_{i-1} = s) = P(s_j = s' \mid s_{j-1} = s), \quad \forall s, s' \in \mathcal{S}, \quad \forall i, j \geq 1.$$

- **Reward function** (R): Designed to minimize congestion, the agent receives negative rewards when vehicle queues exceed critical thresholds and positive rewards when traffic volume is effectively reduced. This function maps state-action pairs to real-valued rewards, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The reward function is also stationary, meaning that $r_i = r_j$ whenever $s_i = s_j$ for all i, j.

- **Discount factor** (γ): A value between 0 and 1 that determines the importance of future rewards in the agent's decision-making process.

The agent aims to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maps states to actions. The objective is to identify a policy that maximizes the expected cumulative discounted reward, thereby improving traffic flow in both directions while minimizing the total number of vehicles queued.

3 Implementation Details

3.1 Environment and Simulator

The project implements a traffic signal control system using a reinforcement learning framework. The system is composed of three components: the **environment**, the **agent**, and the **simulator**.

- **Traffic Simulator:** The simulator models the intersection dynamics by managing vehicle arrivals, departures, and traffic light state transitions. Vehicle arrivals in each direction follow a Poisson distribution, providing a realistic and stochastic representation of traffic flow.
- **Traffic Environment:** This class serves as the interface between the agent and simulator, defining the MDP with states that capture the traffic light status and vehicle counts constrained by maximum limits. The agent's actions consist of switching or maintaining the traffic light. The reward function penalizes congestion and rewards clearing the intersection. Episodes terminate upon reaching a maximum number of steps or when congestion exceeds critical thresholds.
- **Execution Script:** This script facilitates tracking model performance, convergence, and experimental results after training. It integrates the environment, simulator, and agents to run both training and evaluation episodes.

3.2 Reinforcement Learning Algorithms

- **Planners:** Value Iteration and Policy Iteration are implemented with value evaluation and policy improvement steps. Both use the known MDP model to iteratively update values and policies to the optimal solution.
- **Agents:** Q-Learning and SARSA agents are implemented to learn policies through interaction with the environment, without requiring explicit knowledge of transition probabilities. The Q-Learning agent uses off-policy learning, updating Q-values based on the maximum expected future reward, while the SARSA agent employs on-policy updates, following the actions actually taken. Both use epsilon-greedy exploration and update Q-values after each step to progressively improve their policies.

3.3 Design Decisions

- The **state space** is carefully discretized to balance complexity and tractability, focusing on car counts and traffic light states to capture relevant traffic conditions without overwhelming the learning algorithms.
- The **action space** is limited to two discrete actions: switch or hold the green light, simplifying the decision-making process while maintaining meaningful control.
- The **reward function** penalizes congested states and incentivizes clearing the intersection, with experiments conducted to explore different reward shaping strategies and their impact on learning effectiveness.
- **Terminal conditions** reflect safety and operational constraints, ending episodes when critical congestion is reached or after a fixed number of time steps to prevent unbounded training sequences.
- **Hyperparameters** such as discount factor (γ), learning rate (α), exploration rate (ϵ), and convergence thresholds (θ) are set to reasonable defaults and systematically tuned during experimentation.

3.4 Reinforcement Learning Algorithms

3.4.1 Value Iteration

Value Iteration is a model-based dynamic programming algorithm that computes the optimal value function by iteratively updating state values using known transition dynamics and rewards. It is a value-based method that directly derives the optimal policy from the value estimates.

```
1:  Input: MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, threshold $\epsilon$
2:  Initialize: $V'(s) \leftarrow 0$, $V(s) \leftarrow \infty$ for all $s \in \mathcal{S}$
3:  Repeat until $\|V - V'\|_\infty \leq \epsilon$
4:      $V \leftarrow V'$
5:      For each $s \in \mathcal{S}$:
6:      $V'(s) \leftarrow \max_{a \in \mathcal{A}} \left[ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V(s') \right]$
7:  End Repeat
8:  Set: $V^*(s) \leftarrow V(s)$ for all $s \in \mathcal{S}$
9:  Compute: $\pi^*(s) \leftarrow \arg\max_{a \in \mathcal{A}} \left[ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s') \right]$
10: Output: $V^*(s)$ and $\pi^*(s)$ for all $s \in \mathcal{S}$
```

Algorithm 1: Value Iteration (1)

3.4.2 Policy Iteration

Policy Iteration is a model-based algorithm that alternates between policy evaluation and policy improvement using full knowledge of the environment.

```
1:  procedure POLICY ITERATION($M, \epsilon$)
2:  $\pi \leftarrow$ Randomly choose a policy $\pi \in \Pi$
3:  while true do
4:      $V^\pi \leftarrow$ POLICY EVALUATION($M, \pi, \epsilon$)
5:      $\pi^* \leftarrow$ POLICY IMPROVEMENT($M, V^\pi$)
6:      if $\pi^*(s) = \pi(s)$ then
7:      break
8:      else
9:      $\pi \leftarrow \pi^*$
10: $V^* \leftarrow V^\pi$
11: return $V^*(s), \pi^*(s)$ for all $s \in \mathcal{S}$

1:  procedure POLICY EVALUATION($M, \pi, \epsilon$)
2:  For all $s \in \mathcal{S}$, define $R^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) R(s,a)$
3:  For all $s, s' \in \mathcal{S}$, define $P^\pi(s'|s) = \sum_{a \in \mathcal{A}} \pi(a|s) P(s'|s,a)$
4:  For all $s \in \mathcal{S}$, $V_0(s) \leftarrow 0$, $V(s) \leftarrow \infty$
5:  while $\|V - V_0\|_\infty > \epsilon$ do
6:      $V \leftarrow V_0$
7:      For all $s \in \mathcal{S}$, $V_0(s) \leftarrow R^\pi(s) + \gamma \sum_{s' \in \mathcal{S}} P^\pi(s'|s) V(s')$
8:  return $V_0(s)$ for all $s \in \mathcal{S}$

1:  procedure POLICY IMPROVEMENT($M, V^\pi$)
2:  $\hat{\pi}(s) \leftarrow \arg\max_{a \in A} \left[ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^\pi(s') \right], \quad \forall s \in \mathcal{S}$
3:  return $\hat{\pi}(s)$ for all $s \in \mathcal{S}$
```

Algorithm 2: Policy Iteration (1)

3.4.3 Q-Learning

Q-Learning is a model-free, value-based, off-policy algorithm that learns the optimal action-value function by interacting with the environment. It updates estimates using the maximum expected future reward, allowing it to learn the optimal policy independently of the agent's behavior policy.

```
1:  procedure Q-LEARNING($\epsilon, \alpha, \gamma$)
2:  Initialize $Q(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily except $Q(terminal, \cdot) = 0$
3:  $\pi \leftarrow \epsilon$-greedy policy with respect to $Q$
4:  for each episode do
5:          Set $s_1$ as the starting state
6:          $t \leftarrow 1$
7:          loop until episode terminates
8:          Sample action $a_t$ from policy $\pi(s_t)$
9:          Take action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
10:         $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\,(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$
11:         $\pi \leftarrow \epsilon$-greedy policy with respect to $Q$ (policy improvement)
12:         $t \leftarrow t + 1$
13: return $Q, \pi$
```

Algorithm 3: Q-Learning (1)

3.4.4 SARSA

SARSA is a model-free, value-based, on-policy algorithm that updates the action-value function based on the actions actually taken by the current policy. It learns while following and improving its own policy, reflecting the effect of exploration during training.

```
1:  procedure SARSA($\epsilon, \alpha_t$)
2:  Initialize $Q(s, a)$ arbitrarily, except $Q(terminal, \cdot) = 0$
3:  $\pi \leftarrow \epsilon$-greedy policy with respect to $Q$
4:  for each episode do
5:          Set $s_1$ as the starting state
6:          Choose action $a_1$ from policy $\pi(s_1)$
7:          loop until episode terminates
8:          Take action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
9:          Choose action $a_{t+1}$ from policy $\pi(s_{t+1})$
10:         $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t\,(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
11:         $\pi \leftarrow \epsilon$-greedy policy with respect to $Q$ (policy improvement)
12:         $t \leftarrow t + 1$
13: return $Q, \pi$
```

Algorithm 4: SARSA (1)

4 Experimental Setup

4.1 Simulation Settings

The simulation models a four-way intersection with two alternating traffic flows: North-South (NS) and East-West (EW). Unless otherwise specified, the parameters used are shown in Table 1.

4.2 Performance Metrics

Policy effectiveness was assessed using the metrics shown in Table 2.

Table 1: Simulation Parameters for Traffic Intersection Modeling

Poisson arrival rate (λ):	Vehicle arrivals in both the NS and EW directions were modeled using a Poisson distribution with a rate of $\lambda = 2.0$.
Maximum steps per episode:	Each simulation ran for up to 1000 time steps.
Number of trials:	Each experiment was repeated 15 times to ensure reliable results.
Vehicle throughput limit:	A maximum of 5 vehicles were allowed to pass through the intersection per time step when the light was green.

Table 2: Metrics Used for Evaluating Intersection Congestion

threshold_violations	Steps where total cars $\geq N$ or any direction $\geq M$.
average_counts	Mean car count at the intersection over the duration of each episode.
below_thresholds	Longest continuous period during which the intersection remained below both N and M.

4.3 Baseline and Trial Variants

Multiple experimental variants were evaluated against a baseline configuration. Baseline:

- **Traffic settings**: $\lambda_{NS} = 2.0$, $\lambda_{EW} = 2.0$, $M = 15$, $N = 20$, cars leaving per step = 5
- **Reward function**:
  - -100 penalty if total cars $\geq N$, or if NS or EW $\geq M$
  - $-1\times$ total cars
  - $-2\times$ overflow beyond 8 cars in NS or EW
  - $+10$ bonus if the intersection is cleared, total cars = 0
- **Termination condition**:
  - Terminate if NS $\geq M + 5$, EW $\geq M + 5$, total $\geq N + 10$, or both NS ≤ 3 and EW ≤ 3
- **Hyperparameters**:
  - Discount factor: $\gamma = 0.75$
  - Exploration rate: $\epsilon = 1.0$, with decay $\epsilon_{\text{decay}} = 0.99$
  - Number of episodes: 2000
  - Convergence threshold: $\theta = 1 \times 10^{-6}$

Variation trials were conducted to evaluate agent performance under different configurations:

- **Trial 1 – Modified Reward Function:**
  -100 if total cars $\geq N$; -50 if NS or EW $\geq M$
  $-1\times$ total cars ; $+10$ for a cleared intersection
- **Trial 2 – Modified Termination Condition:**
  Terminate if NS $\geq M + 5$, EW $\geq M + 5$, total $\geq N + 10$, or NS ≤ 4 and EW ≤ 2
- **Trial 3 – New Traffic Pattern:** $\lambda_{NS} = 3.0$, $\lambda_{EW} = 1.0$
- **Trial 4 – Lower Discount Factor:** $\gamma = 0.1$ for Q-Learning and SARSA
- **Trial 5 – Fewer Training Episodes:** 100 episodes for Q-Learning and SARSA
- **Trial 6 – Aggressive Epsilon Decay:** $\epsilon = 1.0$, $\epsilon_{\text{decay}} = 0.3$
- **Trial 7 – Larger Convergence Threshold θ:** $\theta = 1 \times 10^{-2}$

These trials enabled evaluation of the robustness of learning algorithms across varying reward functions, traffic patterns, and exploration strategies. Understanding how changes in these parameters affect reinforcement learning behavior is crucial for developing algorithms that better adapt to the environment, thereby improving both efficiency and safety. The detailed outcomes of these experiments are presented in the Results section.

5 Results and Analysis

5.1 Policy Evaluation

The performance of four RL algorithms including Value Iteration, Policy Iteration, Q-Learning, and SARSA was evaluated under baseline settings. Model-based methods (Value and Policy Iteration) converge faster (under 60 iterations) by leveraging known transition dynamics (P), efficiently finding near-optimal policies in this simplified traffic control scenario. However, they are less practical for complex or unknown environments without full models.

Model-free methods (Q-Learning and SARSA) learn directly from environment interactions, offering greater flexibility but requiring over 2000 episodes to converge. Q-Learning typically achieves higher average rewards and lower intersection loads than SARSA, likely due to its off-policy nature enabling broader exploration. Results are summarized in the table below.

	Threshold Violations	Average Counts	Below Thresholds
Value Iteration	0.93	10.46	10.27
Policy Iteration	1.13	8.81	7.53
Q-Learning	0.93	8.33	6.67
SARSA	1.33	10.61	12.40

Table 3: Average performance over 15 experimental trials

5.2 Impact of Reward Design

In Trial 1, removing the penalty of $-2\times$ the overflow beyond 8 vehicles in either the North-South or East-West directions caused the model to deprioritize clearing the more congested direction. As a result, the agent required more steps to clear the intersection (see top left) and maintained a higher average number of vehicles at each time step (see top right).

Reward functions that penalized long wait times or high traffic volumes helped the agent maintain steady traffic flow and reduce congestion. However, this sometimes caused the agent to neglect directions with lighter traffic. In contrast, the baseline model rewarded the agent when traffic in any direction exceeded 8 vehicles, encouraging it to clear congested lanes more aggressively and reducing the risk of critical congestion. While this promoted intervention in heavily congested areas, it also supported a more balanced traffic management strategy.

5.3 Impact of Termination Conditions

In Trial 2, the termination criteria were adjusted to require fewer than 4 vehicles in the North–South (NS) direction and fewer than 2 in the East–West (EW) direction, compared to the baseline threshold of fewer than 3 vehicles in both directions. Despite this change, the overall number of steps required to clear the intersection in Trial 2 showed no significant difference from the baseline. However, analysis of simulation logs revealed a consistent bias: the agent in Trial 2 prioritized clearing the NS direction first. This behavior is likely due to the modified termination condition, which made reducing the NS queue more beneficial during training.

Termination conditions that reflect real-time traffic saturation provide a more realistic framework for traffic control. They encourage the agent to proactively reduce queues to avoid triggering early

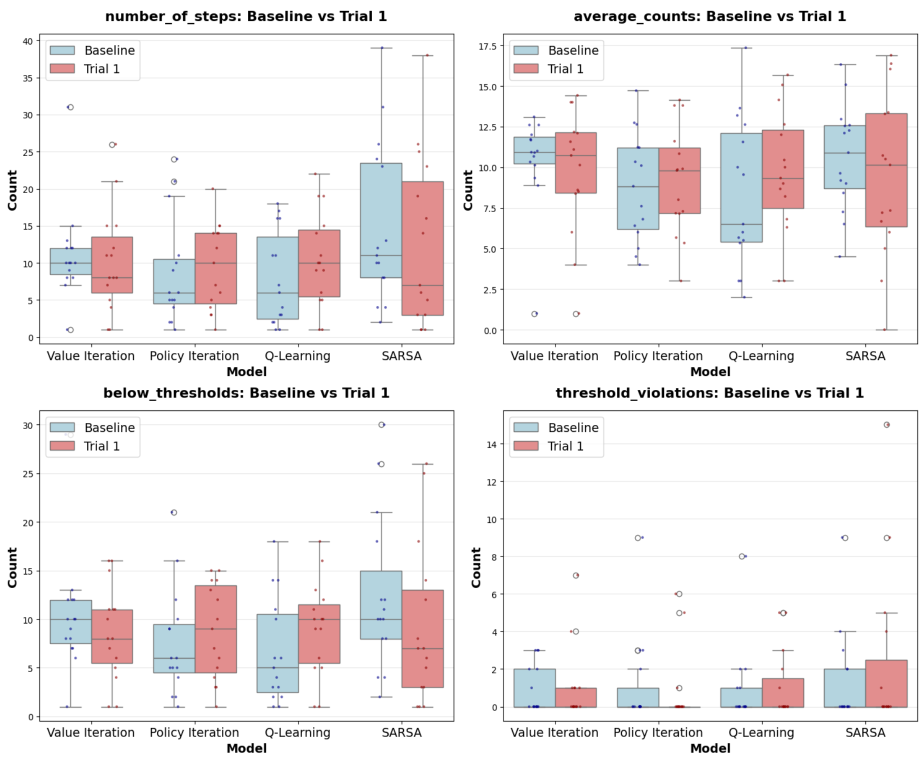


Figure 2: Comparison between Baseline and Trial 1 with a Modified Reward Function.

termination due to congestion. While this approach improves robustness in high-traffic scenarios, it can lead to reduced responsiveness under lighter traffic, as the agent may become overly cautious, delaying signal transitions or prolonging episode duration unnecessarily.

5.4 Impact of Traffic Patterns

In Trial 3, the environment was configured with an imbalanced traffic distribution, where the North–South (NS) direction (with $\lambda_{NS} = 3$) and lower rates in the East–West (EW) direction (with $\lambda_{EW} = 1$). This setup contrasts with the baseline configuration, where both directions had balanced traffic inputs ($\lambda = 2$).

While the agent adapted to the more congested NS direction by prioritizing traffic flow accordingly, the overall performance of the trial with the imbalanced pattern was worse than that of the baseline model. This decline in performance is likely due to the difficulty in accurately estimating traffic demand when the reward function and termination conditions treat both directions equally. As a result, the agent struggled to balance signal timing effectively, leading to increased average congestion and longer episode durations.

5.5 Sensitivity to Hyperparameters

The influence of key hyperparameters, including the convergence threshold (θ), discount factor (γ), exploration rate (ϵ), and number of training episodes, was systematically examined across Trials 4 through 7. Each trial was designed to isolate and evaluate the effect of a specific parameter.

- **Trial 4:** Modified the discount factor γ
- **Trial 5:** Adjusted the number of training episodes

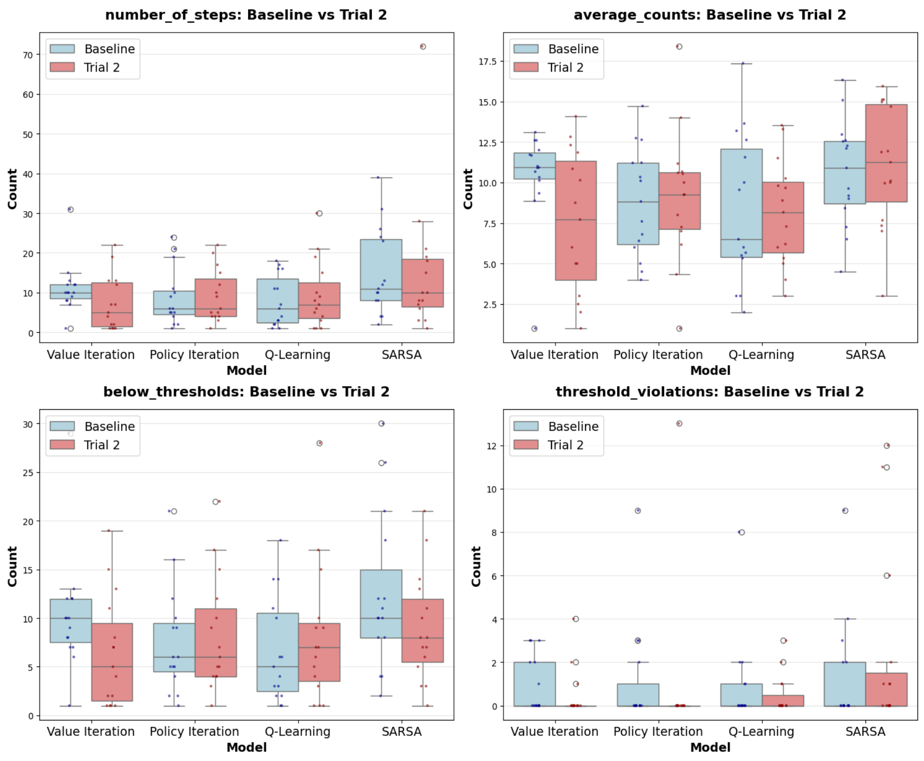


Figure 3: Comparison between Baseline and Trial 2 with a Modified Termination Condition.

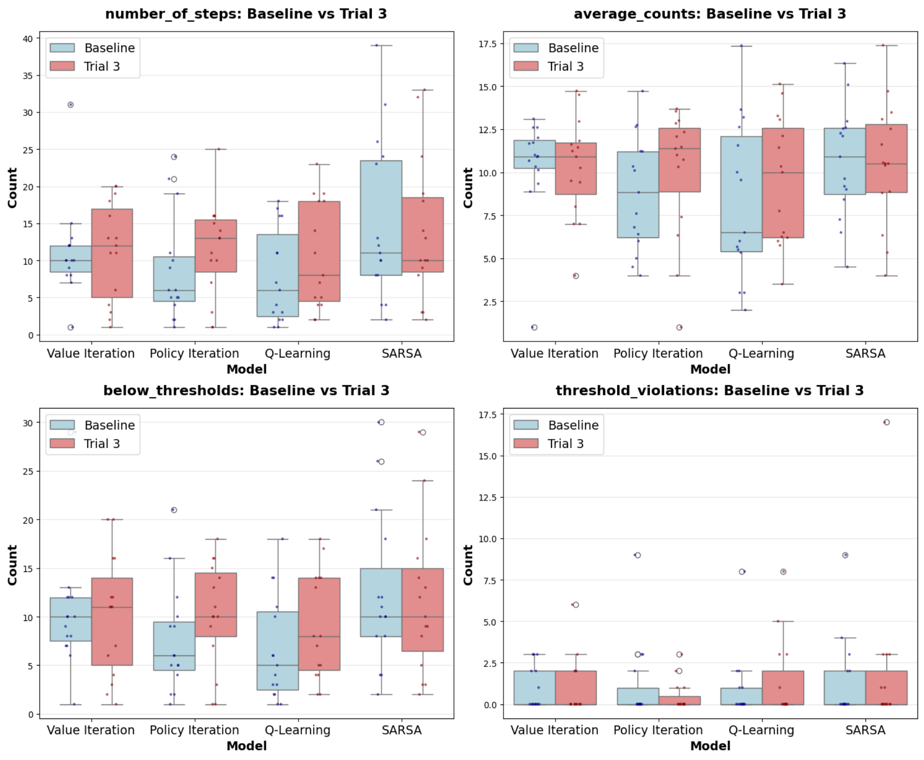


Figure 4: Comparison between Baseline and Trial 3 under Skewed Traffic Patterns.

- **Trial 6:** Altered the exploration rate ϵ
- **Trial 7:** Increased the convergence threshold θ

Convergence Threshold (θ): This parameter defines the minimum change required in the value function between iterations. In model-based methods like Value Iteration and Policy Iteration, a larger θ (e.g., 10^{-2} in Trial 7 versus 10^{-6} in the baseline) causes the algorithm to converge faster. However, this can slightly reduce performance due to less precise value estimation.

Discount Factor (γ): The discount factor determines the weight assigned to future rewards. A high γ encourages long-term planning by valuing future rewards more, while a low γ makes the agent favor short-term gains. In Trial 4, reducing γ from 0.75 to 0.1 degraded performance in scenarios where long-term strategies were beneficial, though it may improve responsiveness in environments where short-term decisions are more critical.

Exploration Rate (ϵ): ϵ balances exploration and exploitation. Higher ϵ promotes exploration, preventing early convergence to suboptimal policies but may cause instability. Lower ϵ favors exploitation but risks premature convergence. Trial 6 used a more aggressive decay ($\epsilon_{\text{decay}} = 0.3$ vs. 0.99 baseline), showing the value of gradually reducing exploration.

Number of Training Episodes: Trial 5 assessed the effect of training duration by reducing the number of training episodes from 2000 (baseline) to 100. A limited number of episodes led to underfitting, reducing the agent's ability to learn effective policies.

Findings: Figure 5 illustrates the impact of hyperparameter variations across the trials. Notably, changes to the discount factor γ in Trial 4 and the training duration in Trial 5 significantly influenced both the convergence speed and the quality of convergence in Q-Learning and SARSA. A change in epsilon decay speed (Trial 6) also affected the final Q-values, as measured by the L_2 norm of the Q-table. While this variance may appear smaller compared to the others, it is important to recognize that the number of states and actions in this simulation is relatively small, and the environment's stochasticity is limited compared to real-world applications. Therefore, although the performance change due to epsilon decay was not significantly large, it remains an important factor influencing the agent's overall performance.

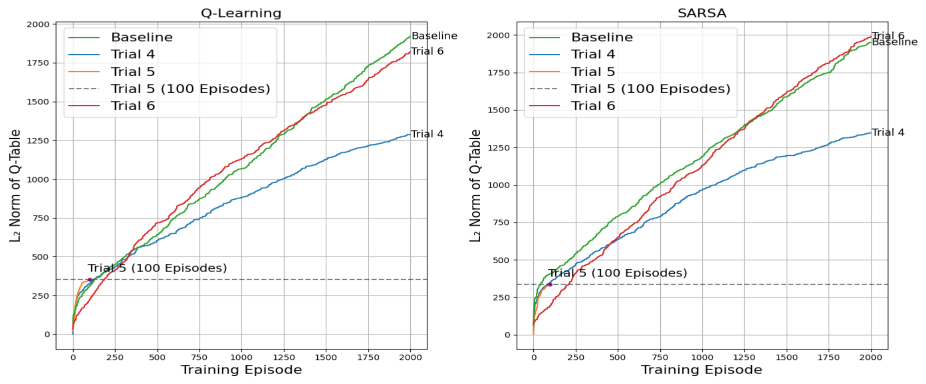


Figure 5: Impact of Hyperparameter Variations on Q-Learning and SARSA Convergence

Figure 6 illustrates the model performance of Value Iteration and Policy Iteration under different values of θ (e.g., 10^{-2} in Trial 7 versus 10^{-6} in the baseline). As expected, Trial 7 converged earlier, requiring fewer than 50 iterations on average, while the baseline model converged in approximately 80 iterations. Due to this early convergence, the performance of the agent in Trial 7 is worse than that of the baseline model, as shown in the figure. Specifically, it exhibits higher variance, a longer average number of steps to reach the terminal state (top right), and a higher average step count overall. This evidence suggests that the agent in Trial 7 is undertrained.

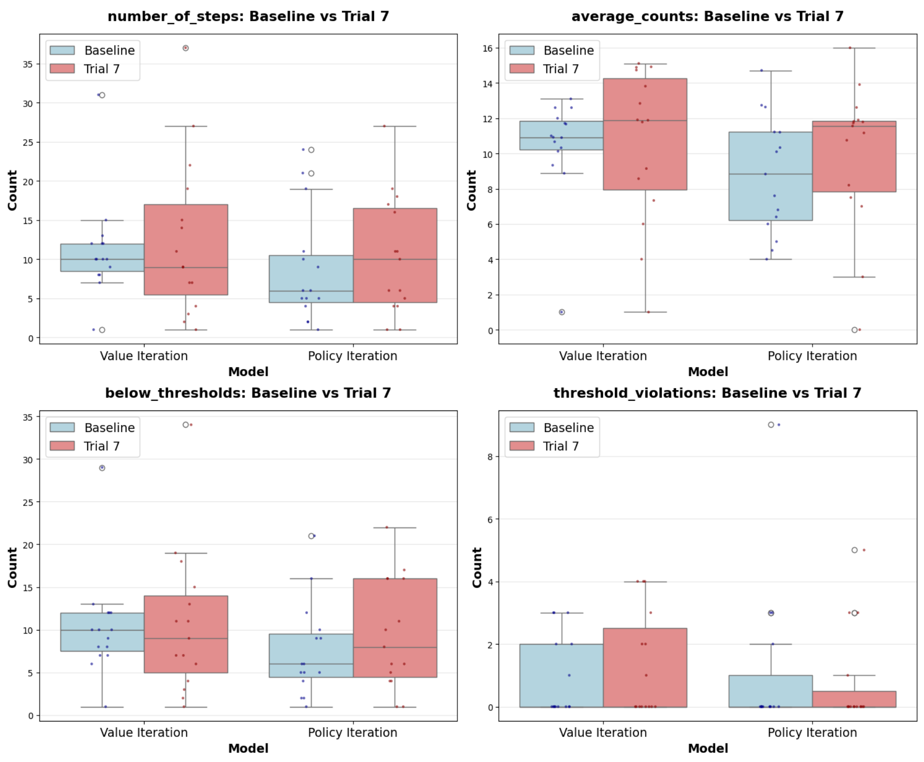


Figure 6: Effect of Convergence Threshold (θ) on Value and Policy Iteration Performance

6 Discussion and Conclusion

This research project explored the use of reinforcement learning (RL) techniques for traffic signal control at a simplified four-way intersection. By applying and comparing Value Iteration, Policy Iteration, Q-Learning, and SARSA, practical insights were gained regarding the performance, limitations, and design considerations of RL-based traffic management systems.

6.1 Main Takeaways

Algorithm Performance and Trade-offs: Model-based approaches such as Value Iteration and Policy Iteration demonstrated fast convergence, often requiring fewer than 60 iterations due to reliance on known environment dynamics. Model-free methods including Q-Learning and SARSA provided greater adaptability in environments with incomplete information. Among these, Q-Learning consistently outperformed SARSA in terms of average reward and traffic throughput, benefiting from its off-policy learning that enables more effective exploration.

Reward Function Design: The reward structure played a critical role in shaping agent behavior. The baseline reward, which penalized overflow beyond eight vehicles per direction, effectively encouraged early congestion mitigation. Removal of this penalty in Trial 1 led to less responsive agents and poorer performance. This outcome highlights the importance of designing reward functions that align closely with traffic management objectives.

Termination Criteria as Behavioral Drivers: Adjusting termination conditions to trigger earlier stopping when queues exceeded certain lengths resulted in more conservative agent policies, reducing extreme congestion at some cost to throughput. Termination thresholds function as indirect behavioral constraints and can guide learning toward safer traffic control strategies.

Sensitivity to Traffic Patterns: Agents generally adapted well to varying traffic conditions but exhibited performance degradation under imbalanced flow scenarios, as observed in Trial 3. This indicates that static reward and termination criteria may not be sufficient in environments with asymmetric traffic demand. Adaptive mechanisms, such as direction-specific penalties or dynamic thresholds, could help maintain effectiveness across diverse scenarios.

Hyperparameter Impact: Trials 4 through 7 demonstrated that hyperparameters such as exploration decay, discount factors, and training duration significantly influence learning efficiency and final policy quality. Aggressive exploration decay or insufficient training episodes led to unstable or short-sighted behaviors. These findings emphasize the necessity of systematic hyperparameter tuning for reliable real-world application.

6.2 Scaling to Real-World Intersections

Although this research project focused on a simplified intersection, deploying RL in real traffic environments requires addressing multiple additional complexities.

Multi-Agent Coordination: Urban traffic networks consist of interconnected intersections. Coordinating control across multiple agents necessitates multi-agent reinforcement learning (MARL) frameworks that support information sharing and cooperative optimization of both local and system-wide traffic flow.

Richer State Representations: Real intersections involve a variety of contextual factors such as pedestrian activity, turning lanes, emergency vehicle priorities, weather conditions, and time-of-day effects. Capturing these complexities requires scalable state representations that may include continuous or hierarchical features.

Expanded Action Spaces: Traffic signal control extends beyond simple binary switching. Controllers must manage phase sequences, signal timing durations, and offsets between intersections. Addressing these requirements will demand more expressive action spaces and potentially hierarchical control policies.

Stochasticity and Uncertainty: Traffic dynamics in real environments are inherently stochastic, influenced by variable vehicle arrivals, driver behavior, and external disturbances. Incorporating stochastic transitions, probabilistic policies, and robust learning algorithms is essential for consistent and reliable performance.

This research project establishes a foundational understanding of reinforcement learning applied to traffic signal control and identifies critical challenges for future research. Insights into algorithm behavior, reward design, and system sensitivity provide valuable guidance for advancing intelligent transportation systems toward real-world implementation.

References

[1] Emma Brunskill. Cs234: Reinforcement learning. https://web.stanford.edu/class/cs234/, 2025. Accessed: 2025-06-08.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.