Project 1 - CS7642: Reinforcement Learning and Decision Making

Syed Muhammad Husainie
`shusainie3@gatech.edu`
GitHub Hash: 402f009
Georgia Institute of Technology
OMSCS – CS 7643: Deep Learning

Abstract

In this project, a reinforcement learning-based traffic signal controller is implemented for a 4-way intersection. The goal of the agent in this case is to select the most optimal policy, ensuring smooth flow of traffic, reduced congestion, and improved safety. The traffic is modeled using a simplistic discrete environment where the state is represented by the number of cars in the north-south and east-west directions, and the signal in the north-south direction. Algorithms including value iteration, policy iteration, Q-learning, and SARSA were implemented and evaluated using custom metrics including average cars waiting per step, policy heatmaps, convergence speed, and return from the policy. The results show that value iteration performed well within the scope of this project, while policy iteration fell short of generating an effective policy to run through 1000 steps of experimentation. SARSA and Q-learning also showed effective implementations, with both showing consistent stability throughout the 1000 steps.

1 Markov Decision Process (MDP) for Traffic Intersection

1.1 State space

The state space for this project was defined by three main parameters: the number of cars in the north-south direction, the number of cars in the east-west direction, and the signal state (green or red) in the north-south direction—which also implicitly defines the signal in the east-west direction. The number of cars in each direction was modeled using a Poisson distribution to simulate realistic traffic arrival patterns.

1.2 Action space

The action space consists of toggling the traffic signal in the north-south direction, which implicitly determines the signal in the east-west direction due to the intersection's constraints. A green signal allows traffic flow in the corresponding direction. The duration of the green signal is governed by a maximum flow threshold, ensuring that it remains green long enough for a meaningful number of cars to pass. While the signal does not switch immediately after the threshold is met, this mechanism enables the environment to reduce congestion in a specific direction, promoting a smoother overall flow of traffic.

1.3 Transition Dynamics

The transition probability matrix constructed for the environment defines the likelihood of moving from one state to another based on the action taken. Each state is represented as a tuple $(ns, ew, light)$, where `ns` denotes the number of cars in the north-south direction, `ew` the number in the east-west direction, and `light` the current traffic light state. For each state, the environment evaluates two actions: maintaining the current light or switching it. For each action, the simulation accounts for cars arriving and departing in both directions, and computes the resulting next state accordingly.

The reward for each transition is calculated, and the simulation checks whether the resulting state is terminal—according to predefined rules that determine episode termination. All observed transitions are collected and normalized such that the probabilities for each state-action pair sum to one. The resulting transition matrix is then used to model the environment's dynamics for reinforcement learning algorithms.

1.4 Reward Function

Three reward functions were tested to evaluate policy performance:

1. **Queue and wait time penalties:** Penalizes both the number of waiting cars and their cumulative wait time, encouraging reduction of congestion and delays. Coefficients were tuned to study policy effects.
2. **Unshaped reward:** Simply penalizes the total number of cars in the system, providing less detailed feedback to assess learning impact.
3. **Light-based incentive:** Adds a small positive reward for cars clearing the intersection during green lights, promoting timely signal changes and improved flow.
4. **Terminal penalty:** A fixed penalty of -10 was applied at terminal states to discourage premature or undesired endings.

1.5 Initial Hyperparameters

Value/Policy Iteration: $\gamma = 0.9$, stopping threshold $= 10^{-6}$.

Q-Learning & SARSA: $\gamma = 0.75$, $\alpha = 0.5$, initial $\epsilon = 1.0$ with decay 0.99, 2000 training episodes.

Environment: queue cap 20 per direction, 30 overall; Poisson rates $(\lambda_{\text{NS}}, \lambda_{\text{EW}}) = (2, 3)$; max departures = 5 cars/step; episode limit = 1000 steps.

These values served as baselines and were later varied to study their impact on learned policies.

1.6 Assumptions in Simplifying the Traffic Intersection

To model the intersection as a discrete Markov Decision Process, several simplifications were made:

- **Simplified intersection:** Modeled a four-way intersection with only north-south and east-west traffic; turns and pedestrians were excluded.
- **Binary signals:** Only red or green phases per direction; no yellow or transitions.
- **Discrete simulation:** All events (arrivals, departures, signal changes) occurred at fixed time steps.
- **Poisson arrivals:** Vehicles arrived randomly at a constant rate with no time-dependent variation.
- **Flow and capacity limits:** Departures were capped per green light, and each direction had a maximum queue length.
- **Instantaneous control:** Signal changes took effect immediately with no delay.
- **Stationary dynamics:** Traffic patterns remained constant throughout the simulation.

1.7 Traffic Signal Implementation

The traffic signal control problem was implemented using several modular Python files:

- **traffic_simulator.py**: Simulated vehicle arrivals (modeled using a Poisson distribution) and departures at each time step. It also tracked how long cars remained in each direction to support a queue-based reward function.
- **traffic_environment.py**: Implements the environment as a discrete MDP, defining state and action spaces, rewards, and transition dynamics. Serves as the interface between the simulator and learning agents.
- **rl_planners.py**: Contains model-based methods—Value Iteration and Policy Iteration—for solving the MDP using known transition and reward models. Additional logging was added to monitor convergence behavior and policy updates.
- **rl_agents.py**: Implements model-free methods, including Q-Learning and SARSA. These agents learn through direct interaction with the environment, adjusting their policies based on observed rewards.
- **traffic_execution.py**: Coordinates the simulation pipeline, executing experiments, collecting results, and generating performance visualizations through tables and plots.

2 Implementation and Discussion of Results

2.1 Impact of reward structure on optimal policy of value iteration

2.1.1 Structured Reward with Queue Length (0.1) and Wait Time (0.9) Penalties

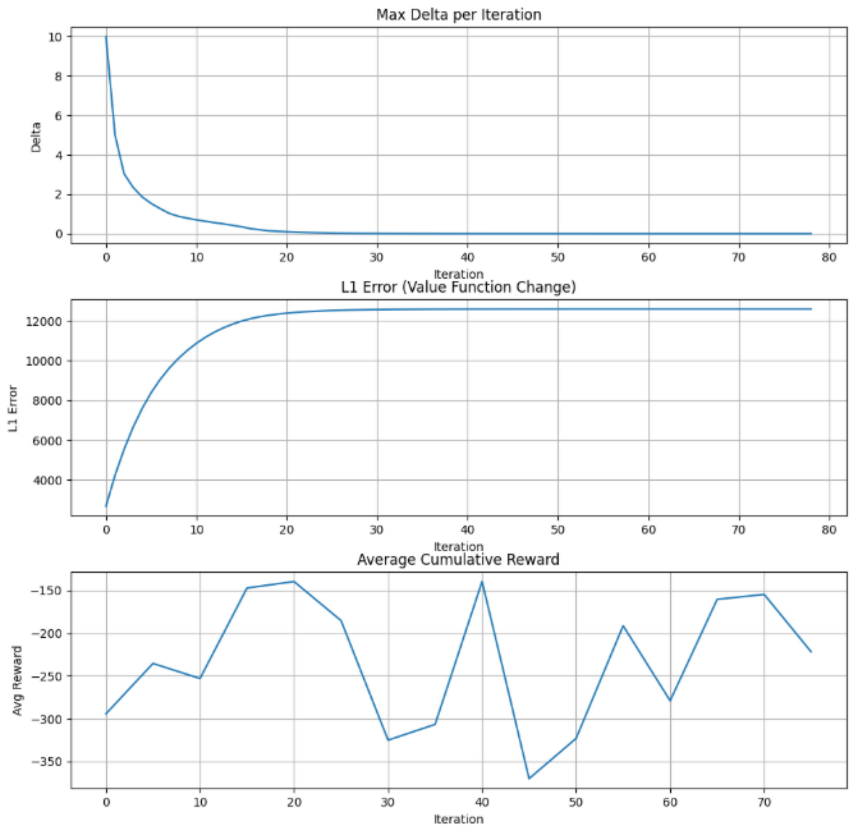


Figure 1: Value iteration convergence for the first reward structure (1.4.1)

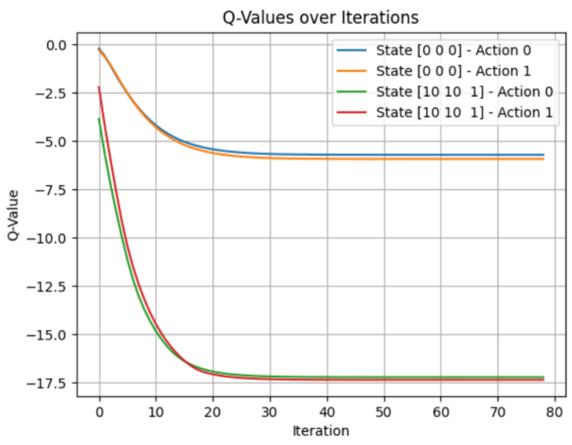
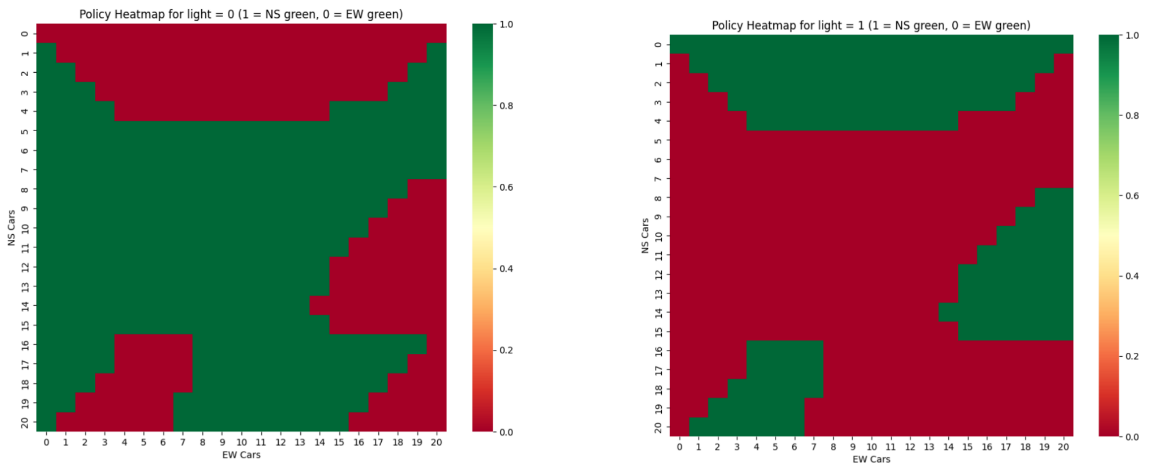


Figure 2: Q-value convergence for two representative states using the original value iteration policy



(a) EW green signal

(b) NS green signal

Figure 3: Policy heatmaps for reward condition 1.4.1 using value iteration

The original reward structure penalized both queue length and wait time. Figures 1 and 2 illustrate that the value iteration policy converged within the first 4–5 steps, as seen from the delta and L1 value changes, which stabilize after about 20 iterations. Despite this, the average cumulative reward fluctuated significantly, suggesting that the learned policy may be suboptimal. This instability likely stems from the reward function itself and the uneven traffic distribution. While the policy appears to aim for long-term balance, with rewards oscillating around -250—this behavior isn't fully captured over just 1000 steps. Perhaps, experimenting over move time steps can capture reward convergence, but this was not investigated for the purpose of this project. To further verify convergence, the Q-values were tracked for two representative states, as shown in Figure 2.

Figure 3 presents the policy heatmaps for scenarios where the green signal is in the east-west and north-south directions, respectively. In Figure 3a, the policy favors a red signal for the east-west direction when the number of cars in the north-south direction is fewer than 4. Additionally, when the number of cars in the east-west direction is either above 15 or below 6—and the north-south direction has approximately more than 17 cars, the policy also prefers a red signal for east-west. Conversely, the policy behaves in a mirrored manner for the north-south direction, as shown in Figure 3b. A possible reason for the behavior in Figures 3a and 3b is the heavily weighted penalty on wait time (0.9) compared to queue length (0.1). This encourages the policy to prioritize reducing how long vehicles wait, rather than how many are present in a queue at any moment. For example, when the number of cars in the north-south direction is fewer than 4, the wait time penalty is likely minimal. As a result, the policy often assigns a red signal to north-south, focusing green time on the east-west direction, even if that queue is relatively long, since the cumulative wait time there may be higher. Similarly, when east-west cars are very high (e.g., >15) or very low (<6), and north-south has a high count (>17), the long wait times in the north-south direction become more significant, leading the policy to prioritize that direction instead. In essence, the policy's decisions are shaped more by how long cars have been waiting than by how many there are, which likely explains the asymmetric prioritization observed in the heatmaps.

2.2 Comparing reward structures

Reward Condition	**Episode Duration**	**Cumulative Reward**	**Avg Cars Waiting**
1.4.1 (wait: 0.9, queue: 0.1)	1000	-1675.1	16.75
1.4.1 (wait: 0.5, queue: 0.5)	522	-4216.5	16.18
1.4.1 (wait: 0.1, queue: 0.9)	522	-7611.1	16.24
1.4.2 (unstructured reward)	522	-8435	16.16
1.4.3 (1.4.1 with light-based incentive)	1000	-789.9	14.47

Table 1: Comparison of reward structures based on episode duration, cumulative reward, and average queue size

The reward structures demonstrate that structuring the reward function influences the policy learned. Only reward conditions 1.4.1 and 1.4.3 completed the full 1000-step episode duration, while the others terminated after 522 steps. Furthermore, reward structure 1.4.3 achieved the highest cumulative reward, suggesting that incorporating a small positive reward for clearing traffic, alongside a significant negative penalty for cars waiting too long at the intersection, leads to the optimal policy—also reflected in the lowest average number of cars waiting. This improved performance likely arises because the positive reward incentivizes the system to actively reduce congestion, while the strong penalty discourages prolonged waiting times, creating a balanced reward signal that encourages efficient traffic flow.

2.3 Effect of terminal conditions

A few terminal conditions were tested to evaluate their efficacy with the value iteration method (using reward function 1.4.3):

1. Terminate the episode if the number of cars in the north-south or east-west direction exceeds the maximum allowed.
2. Terminate the episode if both intersections are clear (i.e., no cars waiting).
3. Terminate the episode if both intersections are clear, or if the wait time in either direction exceeds a threshold (values from 1 to 30 time steps were tested).
4. Terminate the episode if both intersections have cleared traffic to fewer than 3 cars in total.

Terminal Condition	**Episode Duration**	**Cumulative Reward**	**Avg Cars Waiting**
1	Not reached	Not reached	Not reached
2	1000	-789.9	14.47
3	Not reached	Not reached	Not reached
4	131	-115.9	16.47

Table 2: Performance metrics for different terminal conditions

The results showed that clearing the intersection was not a viable terminal condition given the preset Poisson distribution. With a lambda greater than 2 in either direction, traffic remained consistently busy, preventing the terminal condition from being reached. However, setting the terminal condition to require a minimum of 3 cars in total did allow the policy to reach terminal states and learn effectively. Despite this, the results indicated a poorer policy performance compared to the original condition (e.g., terminal condition 2).

2.4 Traffic patterns using Poisson distribution

#	Poisson Distribution & Terminal Condition	Episode Duration	Cumulative Reward	Avg Cars Waiting
1	Uniform rate of 3 in both directions with terminal condition 2	7	-18	27.29
2	Uniform rate of 10 with terminal condition 2	1	-10	38.00
3	Rate of 2 for north-south, rate of 3 for east-west, terminal condition 2 with an added terminal condition of wait time for car not exceeding 2	1000	-428.6	6.92

Table 3: Policy performance for different Poisson traffic rates and terminal conditions.

The experiments with different Poisson traffic rates and terminal conditions reveal significant impacts on policy performance. When using a uniform arrival rate of 3 cars in both directions with terminal condition 2 (episode ends if both intersections clear), the episode duration was very short (7 steps), with a relatively high average queue length (27.29 cars) and a cumulative reward of -18, indicating poor traffic flow management. Increasing the uniform rate to 10 led to an even shorter episode (1 step) with worse congestion (38 cars waiting), showing that heavier traffic under this terminal condition prevents effective learning.

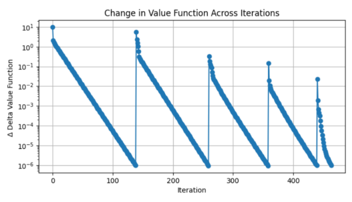
Conversely, introducing a mixed traffic rate (2 cars north-south and 3 cars east-west) combined with terminal condition 2 and an additional terminal condition limiting wait time to 2 steps resulted in a much longer episode duration (1000 steps), a substantially improved cumulative reward (-428.6), and a significantly reduced average queue size (6.92 cars). This suggests that incorporating wait time constraints alongside clearing conditions helps the policy learn a more effective traffic control strategy that better balances traffic flow and reduces congestion.

2.5 Comparing planners and agents

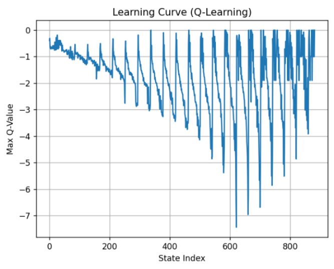
The terminal condition 2 along with reward function 1.4.3 were used for policy iteration, Q-learning, and SARSA. The figures and table below summarize the findings:

Algorithm	γ	θ	α	ϵ	ϵ **Decay**	**Training Episodes**	**Episode Duration**	**Cumulative Reward**	**Avg Cars Waiting**
Value Iteration	0.3	1×10^{-6}	–	–	–	–	190	-156.3	13.56
Value Iteration	0.5	1×10^{-6}	–	–	–	–	190	-156.1	13.56
Value Iteration	0.9	1×10^{-8}	–	–	–	–	1000	-803	15.05
Value Iteration	0.9	1×10^{-3}	–	–	–	–	1000	-859.7	16.13
Value Iteration	0.99	1×10^{-6}	–	–	–	–	22	-29.7	19.82
Policy Iteration	0.3	1×10^{-6}	–	–	–	–	10	-17.1	17.00
Policy Iteration	0.5	1×10^{-6}	–	–	–	–	17	-27.1	21.76
Policy Iteration	0.9	1×10^{-6}	–	–	–	–	179	-153.1	14.79
Policy Iteration	0.9	1×10^{-8}	–	–	–	–	179	-153.0	14.79
Policy Iteration	0.9	1×10^{-3}	–	–	–	–	179	-153.0	14.79
Policy Iteration	0.9	2×10^{-8}	–	–	–	–	179	-153.0	14.79
Q-Learning	0.75	–	0.1	0.5	0.99	2000	1000	-1992.5	23.20
Q-Learning	0.75	–	0.1	0.1	0.99	2000	1000	-1997.6	21.92
Q-Learning	0.75	–	0.1	0	0.99	2000	612	-1228.7	23.06
Q-Learning	0.75	–	0.1	0.7	0.99	2000	1000	-1998.1	21.98
Q-Learning	0.75	–	0.5	0.1	0.99	2000	1000	-1998.1	22.12
Q-Learning	0.75	–	0.01	0.1	0.99	2000	253	-513.1	23.51
Q-Learning	0.75	–	0.1	0.1	0.5	2000	1000	-1997.6	21.92
Q-Learning	0.75	–	0.1	0.1	0.1	2000	1000	-1997.6	21.92
Q-Learning	0.75	–	0.1	0.1	0.9999	2000	36	-74.4	22.58
Q-Learning	0.75	–	0.1	0.1	0.7	2000	36	-74.4	22.58
Q-Learning	0.9	–	0.1	0.1	0.7	2000	221	-444.7	23.05
Q-Learning	0.75	–	0.1	0.1	0.9	3000	929	-1864.9	23.19
SARSA	0.75	–	0.1	0.5	0.99	2000	1000	-2000	22.08

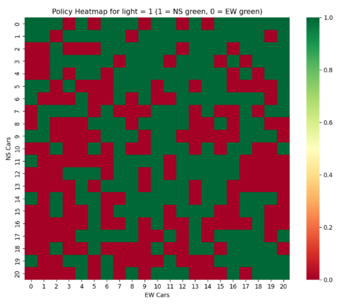
Table 4: Summary of algorithms with their hyperparameters and performance metrics.



(a) Policy iteration convergence



(b) Q-Learning convergence



(c) Q-Learning heatmap

Figure 4: (a) Policy iteration convergence over 5 policies, (b) Q-Learning convergence, and (c) Q-Learning heatmap.

The results from the experiments show that the choice of algorithm and parameter settings play a major role in optimizing traffic control. It was found that Value Iteration with a higher discount factor (e.g., $\gamma = 0.9$) offered the best trade-off between training time and cumulative reward. This suggests that placing more emphasis on long-term rewards helps the algorithm perform better in this environment. In contrast, lower γ values (like 0.3 or 0.5), as well as very high values like 0.99, either underestimated future outcomes or slowed convergence, leading to lower rewards and earlier termination. Policy Iteration did not perform as well, especially at moderate γ values—with episode lengths topping out at just 179 steps, even when using finer θ thresholds. As shown in Figure 4a, it took five policy changes for Policy Iteration to converge, which shows that it adapts more slowly to an optimal policy in this setting.

Model-free methods like Q-Learning and SARSA didn't perform as well compared to Value Iteration. SARSA, in particular, resulted in low cumulative rewards and unstable behavior. Q-Learning was more stable in terms of episode length as it consistently completed all 1000 steps across a variety of hyperparameter settings, but it still did not reach the reward levels of Value Iteration. Figure 4b shows that Q-Learning convergence is steady, but ultimately leads to a suboptimal policy. The policy heatmap in Figure 4c shows a more sporadic decision making pattern for Q-Learning, which highlights its lack of structure in the learned policy for traffic light control. Even though it runs to completion, it does not necessarily make the best decisions.

Overall, these findings highlight the importance of tuning discount factors carefully and show that dynamic programming approaches—especially Value Iteration—are more effective for this type of traffic control problem when the environment's dynamics are known. Lastly, the figures 4 a-b show the convergence patterns for policy iteration and Q learning respectively.

2.6 Conclusions

To summarize, value iteration with terminal condition 2, reward function 1.4.3, and traffic pattern 3 achieved the optimal policy for the traffic control problem. For future trials, it would be interesting to implement some of the assumptions made, to develop a more complex problem.