TRAFFIC SIGNAL CONTROL

——An optimization based strategy

Adheesh Trivedi, Sambit Sahoo, Aditya Sinha

IISER Bhopal

24/04/2025

AdhTri001/traffic-light-optimization

Basic formulation of Traffic light control

Self-Organizing Traffic Lights (SOTL)

Simulation Specification

Optimization for parameters

RL Approaches

Basic formulation of Traffic light control

Self-Organizing Traffic Lights (SOTL)

Simulation Specification

Optimization for parameters

RL Approaches

Waiting time minimization

- The goal is to formulate ‘waiting time’ minimization of vehicles on traffic light as optimization problem. Here we give a most elaborate and straight forward formulation of waiting time minimization.

Waiting time minimization

- The goal is to formulate 'waiting time' minimization of vehicles on traffic light as optimization problem. Here we give a most elaborate and straight forward formulation of waiting time minimization.
- Assume we are working with a single traffic light. Let there be N vehicles, each with waiting time w_i $\forall i = 1, 2, \cdots, N$, during the time frame of T, which is larger than cycle time of traffic light.

Waiting time minimization

- The goal is to formulate 'waiting time' minimization of vehicles on traffic light as optimization problem. Here we give a most elaborate and straight forward formulation of waiting time minimization.
- Assume we are working with a single traffic light. Let there be N vehicles, each with waiting time w_i $\forall i = 1, 2, \cdots, N$, during the time frame of T, which is larger than cycle time of traffic light.
- **Objective function:** Minimize the following

$$\sum_{i=i}^{N} w_i$$

Waiting time minimization

- The goal is to formulate 'waiting time' minimization of vehicles on traffic light as optimization problem. Here we give a most elaborate and straight forward formulation of waiting time minimization.
- Assume we are working with a single traffic light. Let there be N vehicles, each with waiting time w_i $\forall i = 1, 2, \cdots, N$, during the time frame of T, which is larger than cycle time of traffic light.
- **Objective function:** Minimize the following

$$\sum_{i=i}^{N} w_i$$

- There are obvious flaws in this approach!

Problems!

- **Starvation:** By focusing on the total sum, you can drive the average down at the expense of a few vehicles piling up very large waits for minorities.
- **Traffic light control** is not very straight forward to define.
- While the formulation is aiming to minimize net waiting time, we might not be able to assign the waiting time to each vehicle at first place during simulation.

Basic formulation of Traffic light control

Self-Organizing Traffic Lights (SOTL)

Simulation Specification

Optimization for parameters

RL Approaches

What is SOTL control?

- A self-organizing system would be one in which elements are designed to dynamically and autonomously solve a problem or perform a function at the system level.
- Our traffic lights are self-organizing because each one decides based only on local information its own state. Still, they manage to achieve robust and adaptive global coordination.
- This type of implementation for Traffic light control have parameters that can help us to formulate waiting time minimization, with these parameters as variables.

Pseudocode

Variables:

S_i = Traffic light state of i^{th} lane
K_i = No. of vehicles waiting in i^{th} lane
ϕ_i = Time since same state of light
ϕ_{min} = Green light time threshold
ϕ_c = Yellow light time threshold
θ = Threshold of number of vehicles waiting in i^{th} lane, before lane makes request for green light

Pseudocode

```
for i = 1 to n:
    if (S_i = Red):
        if (K_i > θ and i ∉ request):
            request.push(i)
        else if (K_i > 0 and request.empty()):
            request.push(i)
    if (S_i = Green):
        if (φ_i ≥ φ_min and ¬(request.empty())):
            S_i = Yellow Light
            φ_i = 0
    if (S_i = Yellow):
        if (φ_i ≥ φ_c):
            req = request.pop()
            S_req = Green
            φ_i = 0
            S_i = Red
```

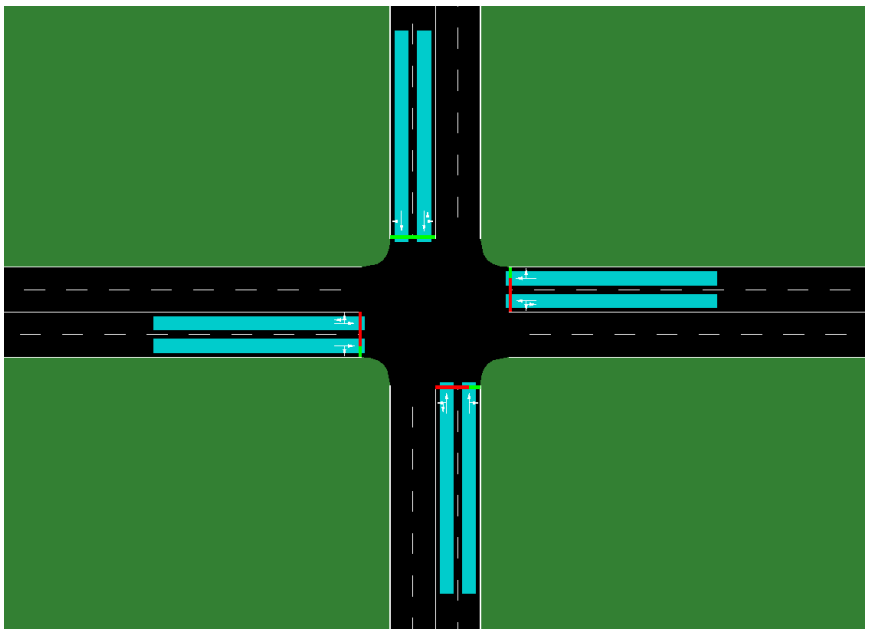
Simulation Software - SUMO

"**S**imulation of **U**rban **MO**bility" is an open source, highly portable, microscopic traffic simulation package designed to handle large networks. It allows for intermodal simulation including pedestrians and comes with a large set of tools for scenario creation. It is mainly developed by employees of the Institute of Transportation Systems at the German Aerospace Center.

4 Lanes test network

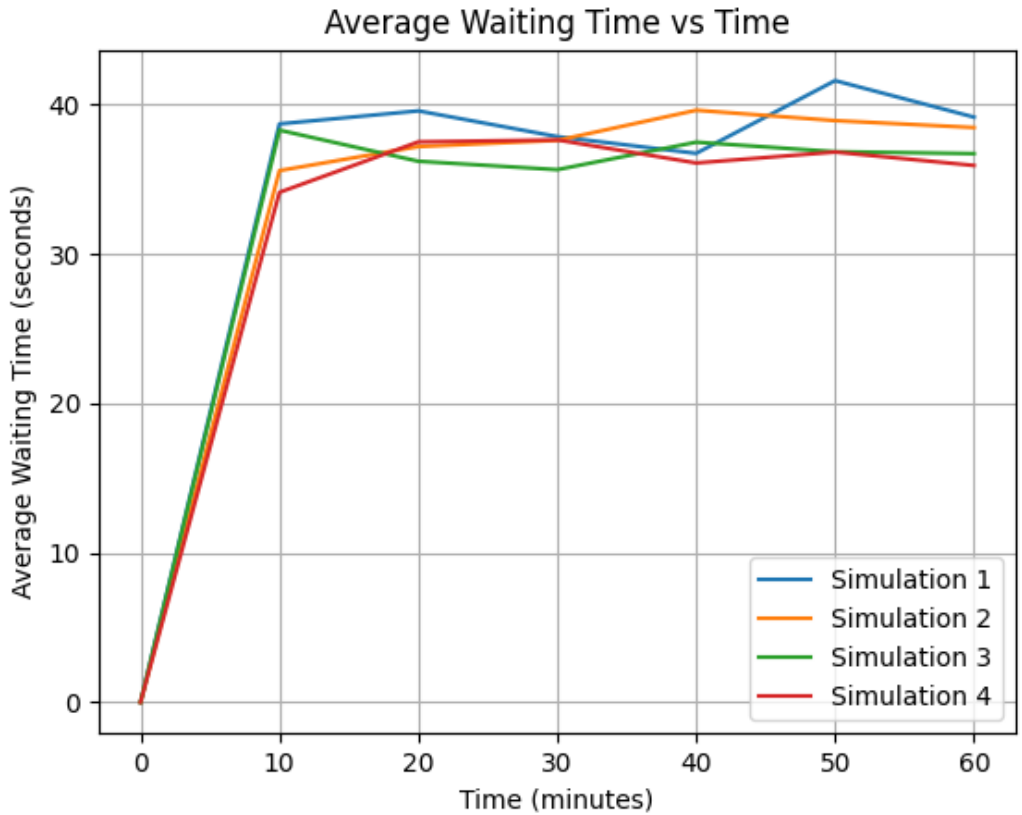
We made the following network, and generated random traffic on it with a fixed number of vehicles per hour on it. There is a traffic light which is controlled by the code using “SOTL-request” described in pseudocode.

The blue lines are vehicle detector areas, which can give count of vehicles at an instant over the area. This can be mimicked in real life by cameras that are usually installed in Traffic lights for detecting traffic violation.



Attempt 1: Fixed green timings

- We did simulation for 250, 350, 450 and 550 vehicles per hour from each lane. Each simulation has fixed green light time of 30 seconds.
- Noticed that the graph is roughly constant for different vehicles per hour. This means that the vehicles are waiting too long then the should.



General Formulation

General Formulation

- Let, there be N lanes, and ϕ_i be the maximum green light time for i^{th} lane.

General Formulation

- Let, there be N lanes, and ϕ_i be the maximum green light time for i^{th} lane.
- Let $a_1, a_2, \cdots, a_N$ be vehicles per second for each lane.

General Formulation

- Let, there be N lanes, and ϕ_i be the maximum green light time for i^{th} lane.
- Let $a_1, a_2, \cdots, a_N$ be vehicles per second for each lane.
- ϕ_{yellow} is a constant, which is the time for which the light is in yellow state.

General Formulation

- Let, there be N lanes, and ϕ_i be the maximum green light time for i^{th} lane.
- Let $a_1, a_2, \cdots, a_N$ be vehicles per second for each lane.
- ϕ_{yellow} is a constant, which is the time for which the light is in yellow state.
- Define, $\psi_i = \sum_{j \neq i} \phi_i + N\phi_{yellow}$ which means the duration i^{th} lane is not green, assuming that all other lanes turn green exactly once before i^{th} lane turn green.

General Formulation

- Let, there be N lanes, and ϕ_i be the maximum green light time for i^{th} lane.
- Let $a_1, a_2, \cdots, a_N$ be vehicles per second for each lane.
- ϕ_{yellow} is a constant, which is the time for which the light is in yellow state.
- Define, $\psi_i = \sum_{j \neq i} \phi_i + N\phi_{yellow}$ which means the duration i^{th} lane is not green, assuming that all other lanes turn green exactly once before i^{th} lane turn green.

Objective function: Our goal is to minimize:

$$\sum_{i=1}^{N} a_i \psi_i$$

General Formulation

- Let, there be N lanes, and ϕ_i be the maximum green light time for i^{th} lane.
- Let $a_1, a_2, \cdots, a_N$ be vehicles per second for each lane.
- ϕ_{yellow} is a constant, which is the time for which the light is in yellow state.
- Define, $\psi_i = \sum_{j \neq i} \phi_i + N\phi_{yellow}$ which means the duration i^{th} lane is not green, assuming that all other lanes turn green exactly once before i^{th} lane turn green.

Objective function: Our goal is to minimize:

$$\sum_{i=1}^{N} a_i \psi_i$$

The idea is that for each lane, we try to minimize the "stacking of vehicles". For i^{th} lane, $a_i\psi_i$ vehicles stack up when the light is not green.

General Formulation

Objective function: Our goal is to minimize:

$$\sum_{i=1}^{N} a_i \psi_i$$

Constraints:

General Formulation

Objective function: Our goal is to minimize:

$$\sum_{i=1}^{N} a_i \psi_i$$

Constraints:

- $\phi_i \geq 5$ prevents the green time becoming to small to be realistic.

General Formulation

Objective function: Our goal is to minimize:

$$\sum_{i=1}^{N} a_i \psi_i$$

Constraints:

- $\phi_i \geq 5$ prevents the green time becoming to small to be realistic.
- $\forall i = 1, 2, \cdots, N$ we add soft constraint $\phi_i \geq 2 a_i \psi_i$. The key idea behind this is, we want to empty all the "stacked up vehicles". But this won't always be possible, so we add it as soft constraint.

General Formulation

Objective function: Our goal is to minimize:

$$\sum_{i=1}^{N} a_i \psi_i$$

Constraints:

- $\phi_i \geq 5$ prevents the green time becoming to small to be realistic.
- $\forall i = 1, 2, \cdots, N$ we add soft constraint $\phi_i \geq 2 a_i \psi_i$. The key idea behind this is, we want to empty all the "stacked up vehicles". But this won't always be possible, so we add it as soft constraint.
- ϕ_i is integer. This becomes an ILP problem.

Formulation for 4 lane

Formulation for 4 lane

- We have 4 lanes, such that $a_1 = a_2 = a_3 = a_4 = a$

Formulation for 4 lane

- We have 4 lanes, such that $a_1 = a_2 = a_3 = a_4 = a$
- This reduced our objective function to just minimize $\phi_i = \phi$ as everything else is becomes constant.

Formulation for 4 lane

- We have 4 lanes, such that $a_1 = a_2 = a_3 = a_4 = a$
- This reduced our objective function to just minimize $\phi_i = \phi$ as everything else is becomes constant.
- So, minimum value of ϕ that will satisfy the constraints is

$$\phi = \max\{5, 2a\psi\}$$

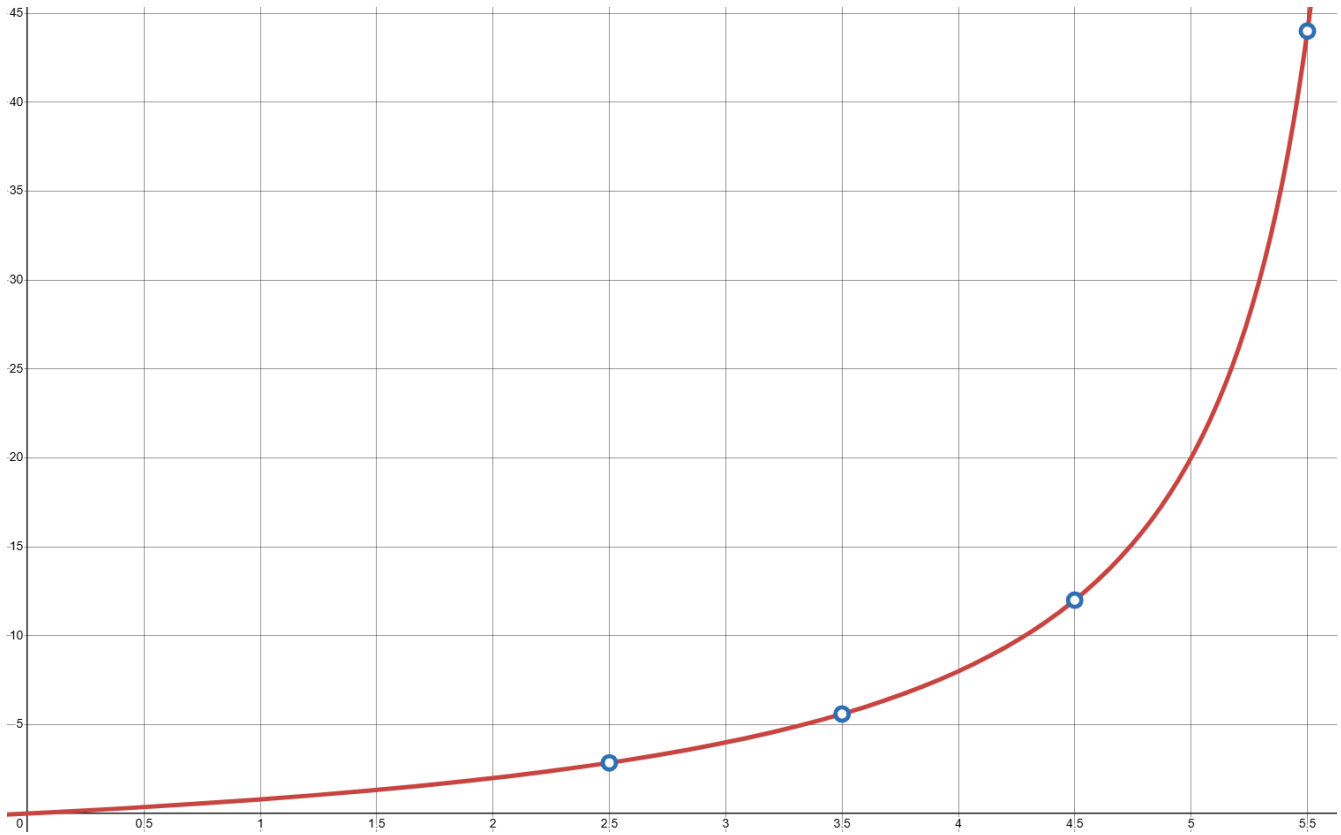
After solving for $\phi_{yellow} = 3$,

$$\phi = \max\left\{5, \left\lceil \frac{24a}{1-6a} \right\rceil\right\}$$

Attempt 2: Optimized Parameters

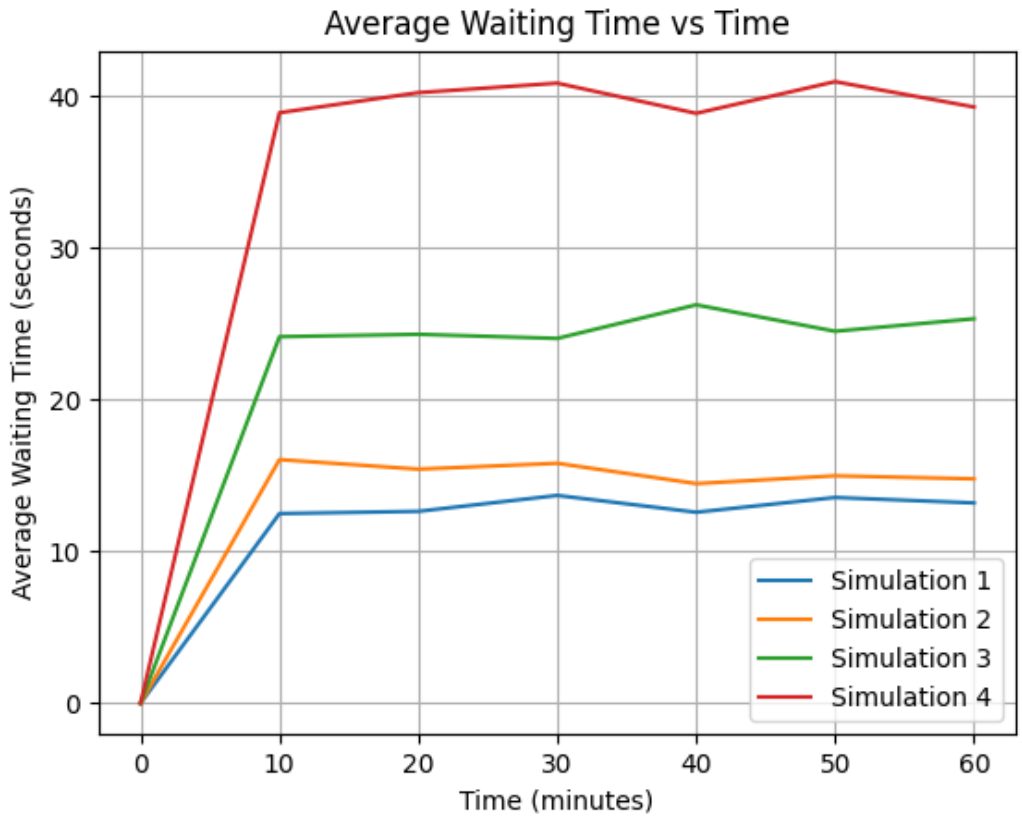
After optimizing the green light timings (ϕ_i) for simulations we got the following results:

a	ϕ
250	5
350	6
450	12
550	44



Attempt 2: Optimized Parameters (Results)

- After simulating for 1 hour duration for each a we got the following result.
- The results are better then unoptimized parameters. As for lesser a the vehicles are waiting much lesser.



RECAP: Fixed-Time Control

- The simplest form of traffic light control.
- Traffic signals operate on a predefined schedule with constant green light durations.
- Does not adapt to real-time traffic conditions.
- Can lead to increased waiting times during congestion or underutilization during sparse traffic.

Q-Learning Based Control

Q-Learning Based Control

- **Model-free Reinforcement Learning (RL)** technique.

Q-Learning Based Control

- **Model-free Reinforcement Learning (RL)** technique.
- **State:** Queue lengths / vehicle densities on each lane.

Q-Learning Based Control

- **Model-free Reinforcement Learning (RL)** technique.
- **State:** Queue lengths / vehicle densities on each lane.
- **Action:** Switch the green light to a particular lane.

Q-Learning Based Control

- **Model-free Reinforcement Learning (RL)** technique.
- **State:** Queue lengths / vehicle densities on each lane.
- **Action:** Switch the green light to a particular lane.
- **Reward:** Negative of total waiting time (or reduction in wait).

Q-Learning Based Control

- **Model-free Reinforcement Learning (RL)** technique.
- **State:** Queue lengths / vehicle densities on each lane.
- **Action:** Switch the green light to a particular lane.
- **Reward:** Negative of total waiting time (or reduction in wait).
- Learns an optimal policy using a Q-table updated via:

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

Q-Learning Based Control

- **Model-free Reinforcement Learning (RL)** technique.
- **State:** Queue lengths / vehicle densities on each lane.
- **Action:** Switch the green light to a particular lane.
- **Reward:** Negative of total waiting time (or reduction in wait).
- Learns an optimal policy using a Q-table updated via:

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

- Suitable for simple environments with discrete state spaces.

Deep Q-Learning (DQL)

Deep Q-Learning (DQL)

- Extends Q-Learning by replacing the Q-table with a Deep Neural Network.

Deep Q-Learning (DQL)

- Extends Q-Learning by replacing the Q-table with a Deep Neural Network.
- Generalizes to complex and continuous state spaces.

Deep Q-Learning (DQL)

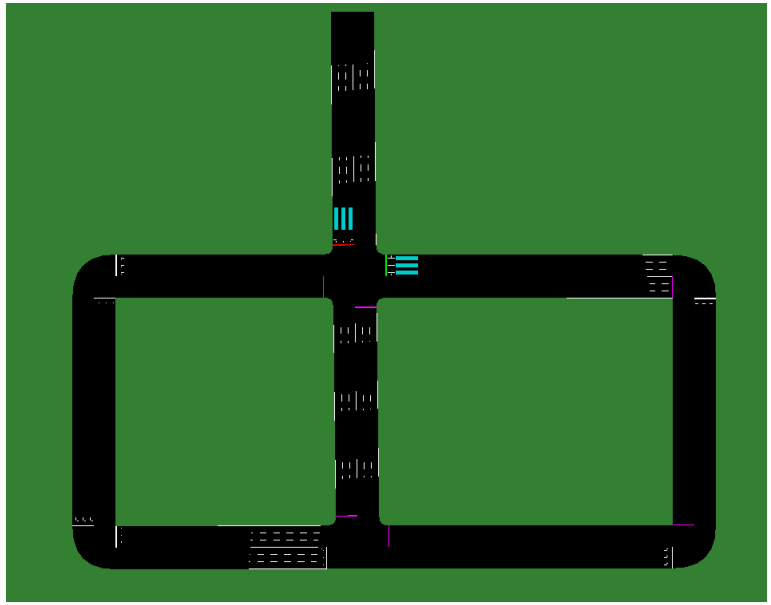
- Extends Q-Learning by replacing the Q-table with a Deep Neural Network.
- Generalizes to complex and continuous state spaces.
- Uses experience replay and target networks for stability:
  - **Replay buffer:** Stores past transitions for batch training.
  - **Target network:** Mitigates oscillations in Q-values during training.

Deep Q-Learning (DQL)

- Extends Q-Learning by replacing the Q-table with a Deep Neural Network.
- Generalizes to complex and continuous state spaces.
- Uses experience replay and target networks for stability:
  - **Replay buffer:** Stores past transitions for batch training.
  - **Target network:** Mitigates oscillations in Q-values during training.
- More robust to dynamic environments and scalable to larger networks.

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

Road Network Used

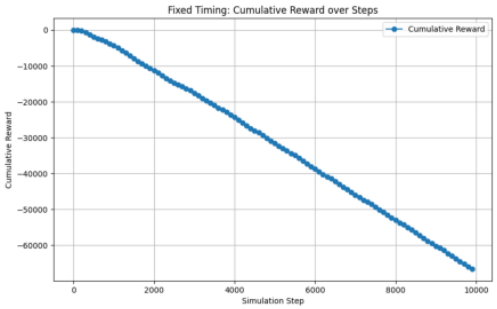


4-Way Intersection Road Network used for simulation in SUMO

- The simulation is based on a complex 4-way intersection with three lanes each.
- Each direction consists of a single incoming and outgoing lane.
- Traffic lights are centrally controlled with fixed, Q-learning, or DQN strategies.
- Vehicles are generated at random intervals to simulate real-world traffic conditions.

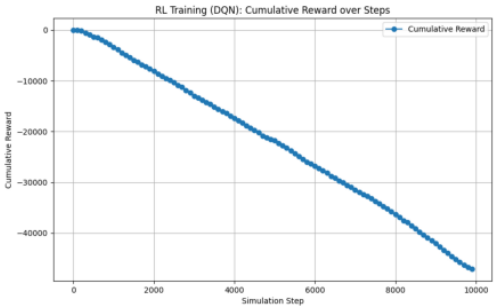
Cumulative Reward Comparison

Fixed Time



Q-Learning

Deep Q-Learning

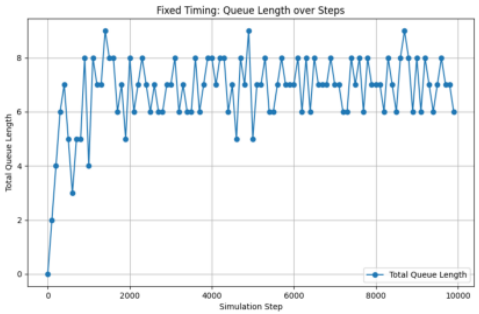


Comparison of cumulative reward across Fixed Time, Q-Learning, and Deep Q-Learning

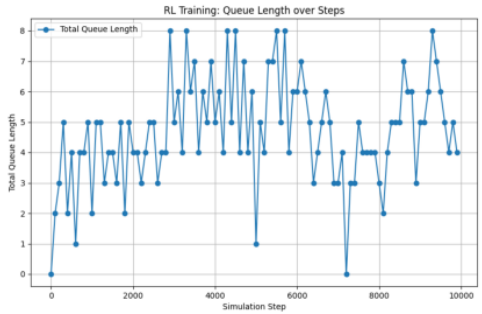
- Fixed Time has the lowest cumulative reward (worst performance).
- Q-Learning improves adaptability but still suboptimal.
- Deep Q-Learning achieves the best reward trend, indicating more efficient learning.

Queue Length Comparison

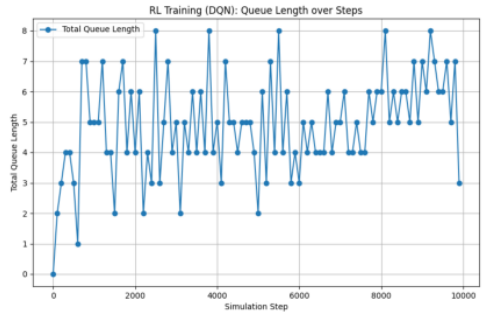
Fixed Time



Q-Learning



Deep Q-Learning



Queue length trends across Fixed Time, Q-Learning, and Deep Q-Learning

- Fixed Time shows consistently high queue lengths.
- Q-Learning begins to reduce queue size via learning.
- Deep Q-Learning maintains lower and more stable queue lengths.

Future Scopes

- Extend the study to account for varying vehicle densities across individual lanes.
- Tesing on huge networks (as shown in the figure).
- Incorporating a variety of vehicle types, such as buses, trucks, and two-wheelers, can enhance the realism and applicability of the traffic model.
- Implement logic for prioritizing emergency vehicles such as ambulances, fire trucks.