Traffic Optimization using Reinforcement Learning: A Decentralized and Hardware-Based Solution

Dr. M. Ramachandra[1] , Mirupuri Aashreya[2] , Chittaloori Rekha[3] , Battini Deekshitha Reddy[3]
Associate Professor, Department of CSE, Sreenidhi Institute of Science and Technology, Hyderabad, India[1]
Student, Department of CSE, Sreenidhi Institute of Science and Technology, Hyderabad, India[2,3,4]

Abstract: **Urban traffic congestion seriously challenges transportation efficiency, safety, and sustainability, usually leading to excessive delays, higher fuel usage, and environmental pollution. In this paper, we introduce a decentralized traffic signal control system using Reinforcement Learning (RL), where every intersection acts as an autonomous agent using tabular Q-learning for decision-making. As opposed to traditional deep learning or centralized methods, ours focuses on simplicity, scalability, and practical applicability. For improving training stability and coverage, we use a pseudo-random event-based simulation method with the SUMO traffic simulator to allow the agent to generalize over diverse traffic patterns. Our system has a minimum green signal duration constraint to mimic real traffic policies and to provide smoother transition. In addition, we couple the model with Arduino hardware to test its viability in real-world settings. Testing on several traffic grid settings shows considerable minimization of vehicle waiting time and enhanced overall traffic flow, revealing the viability of our method for real-world implementations in intelligent transportation systems.**

Keywords—Reinforcement Learning, Traffic Signal Control, Smart Cities, SUMO Simulator, Q-learning, Decentralized Systems, Arduino Integration, Intelligent Transportation Systems, Event-Based Simulation, Real-Time Decision Making.

I. INTRODUCTION

Urban traffic congestion is a chronic problem that has serious effects on commuter productivity, fuel usage, and air quality. Traditional traffic signal systems—usually founded on fixed-timing or elementary sensor-actuated approaches—are not responsive enough to change as quickly as traffic conditions do, thus leading to excessive delays and inefficiencies. Recent developments in Reinforcement Learning (RL) have shown significant promise in making adaptive traffic control possible through the ability of agents to learn the best signal timings by experiencing the environment. In several recent works, RL-based approaches like Deep Q-Networks (DQN) and Multi-Agent RL have been used to control traffic lights and demonstrated improvements in capacity and decreased waiting times. Still, they tend to be based on complicated neural network structures and centralized control mechanisms, which might not be easily scalable or interpretable for huge urban networks. Additionally, such methods often neglect real-world deployment considerations, such as hardware-level integration and compliance with traffic regulation protocols (e.g., minimum green time requirements).

To counter these limitations, we introduce a decentralized framework of traffic light control utilizing tabular Q-learning—a less complex but efficient RL algorithm—for individual intersections. Our approach uses a pseudo-random, event-based training process for stabilizing learning and achieving reproducibility under varied traffic patterns. Moreover, the system imposes a minimum green signal duration to comply with real-world policies and proves its practical feasibility through Arduino-based hardware integration. The model is simulated and validated using the SUMO traffic simulator, which enables us to simulate realistic urban traffic conditions. This paper endeavours to offer an interpretable, scalable, and low-cost solution for intelligent traffic management systems, connecting theoretical research with real-world practice.

In short, we suggest a hardware-compatible, interpretable, and decentralized traffic control solution based on Q-learning.

II. RELATED WORK

RL has been explored for application in traffic systems under various architectures and algorithms. Q-learning, one of the earliest RL algorithms, has been implemented effectively in single-agent traffic intersections. Its variants like Deep Q-Networks (DQN) and Multi-Agent Deep Deterministic Policy Gradient (MADDPG) have been proposed to handle larger, multi-intersection scenarios.

Genders and Razavi (2016) suggested applying DQN to traffic signal control using convolutional neural networks for processing traffic states. Li et al. (2022) suggested a distributed RL algorithm that adjusted signal timing

dynamically. Most of these approaches, however, ignore the problems of implementation complexity, and few provide a real route to hardware deployment.

Compared to these systems, our method focuses on interpretability, modularity, and deployment readiness using tabular Q-learning, Arduino testing, and event-based fixed training. These advancements bridge the gap between theoretical modeling and urban traffic deployment.

III. SYSTEM ARCHITECTURE

Our architecture consists of the following main components:

- Simulation Module: Developed in SUMO, a microscopic traffic simulation tool, giving us fine-grained control over vehicle movement and intersection behavior.
- RL Agent: Implemented using tabular Q-learning, there is one agent per intersection node that executes concurrently.
- State Representation: Figured as the count of cars in every arrival lane.
- Hardware Module: Simulation outputs are read by Arduino microcontrollers and control LEDs used as traffic lights.
- Fixed Event Generator: A deterministic collection of traffic events produced by a script used for reproducible training.

Figure 1 shows the high-level overview of this architecture. The diagram includes the agent-environment interaction loop, SUMO simulation environment, and integration with the hardware interface.

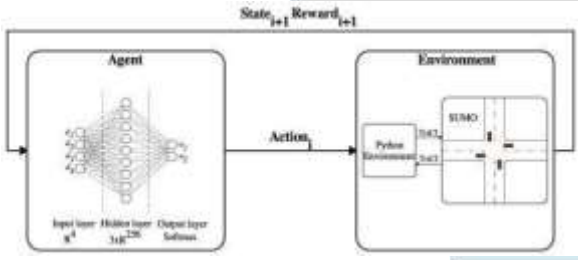

Fig. 1. System Architecture Overview (custom architecture with agent nodes, SUMO simulator, and hardware interface).

IV. METHODOLOGY

A. *Q-Learning Algorithm*

Our method uses the proven Q-learning algorithm, a model-free, off-policy reinforcement learning technique. The Q-value for every state-action pair is updated iteratively with the following rule:

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

where s is the current state, a is the selected action, r is the immediate reward, s is the next state, α is the learning rate, and γ is the discount factor. This iterative update allows the agent to learn an approximation of the optimal action-value function over time, thus refining its decision-making policy for traffic signal control.

B. *State, Action, and Reward*

In our system, state s at each intersection is the four-dimensional vector of vehicle counts on the four entry lanes. The action a is a binary choice for each intersection: which one lane to turn green. The reward r is negative times the product of the total waiting time—vehicles times seconds. This design punishes lengthy waiting and encourages the minimization of delays, with a clear focus on traffic flow efficiency improvements.

C. *Fixed Events Training*

In order to counter the high variance usually experienced with fully random traffic inputs, we take a pseudo-random, fixed-event training approach. A deterministic traffic scenario set is created with a self-written script and repeatedly replayed throughout training. This strategy reduces variance in Q-value updates and improves convergence stability, enabling the agent to generalize more robustly across varied traffic conditions.

D. *Minimum Green Time Constraint*

Adding realistic traffic control restrictions is essential for real-world deployment. Thus, every action—i.e., the choice of a lane for the green light—is held for at least 30 seconds. This restriction avoids too frequent signal switching and ensures smoother changes, thus bringing our model in sync with existing traffic rules.

E. *Arduino Integration*

One of the central elements of our approach is the synthesis of the simulation output with real-world physical hardware. The Q-learning agent's decisions are sent over serial communication to an Arduino Uno, which drives an array of LEDs simulating traffic lights. This hardware-in-the-loop test ensures that our method is not just successful in simulation but also implementable in real-world applications for intelligent transportation systems.

F. *Full Training Model Algorithm*

The training procedure of our decentralized Q-learning agent is described in the following pseudocode:

Algorithm 1: Decentralized Q-Learning Training for Traffic Signal Control

Input: Initial state s_0, maximum episodes K, episode length T
Output: Trained agent (Q-table parameters θ)

1: Initialize Q-table θ with zeros

2: for episode = 1 to K do

3: Initialize state s = s_o

4: for t = 1 to T do

5: for each intersection n in network do

6: Observe state s

7: Choose action a using ε-greedy policy

8: Execute action a

9: Observe reward r and next state s '

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

11: end for

12: end for

13: end for

Output: Trained Q-table

After training over K=500 episodes, each intersection agent converges to a stable Q-table. Below is an excerpt of the learned Q-values for a representative intersection. The agent prefers the direction with heavier traffic, as indicated by higher Q-values associated with green lights in that direction.

Sample Trained Q-table (for a single intersection)

State	Action A0 (NS Green)	Action A1 (EW Green)
(High, Low)	7.8	3.1
(Low, High)	2.2	8.4
(Medium, Medium)	5.0	5.1
(Low, Low)	3.5	3.2
(High, High)	6.9	6.7

State format: (NS vehicle density, EW vehicle density)
Action: A0 = green for North-South, A1 = green for East-West

This table highlights how Q-learning captures traffic patterns and prioritizes actions accordingly to reduce overall vehicle wait times.

This figure describes the entire training process of the RL agent and its integration with hardware afterwards. The process starts with the Fixed Event Generator generating reproducible traffic scenarios, which are subsequently run in the SUMO simulation environment. The RL agent, through Q-learning, modifies its Q-table from the perceived states, actions, and rewards. Lastly, the optimized choices are sent through serial communication to an Arduino-based hardware platform that manages LEDs as traffic lights

V .IMPLEMENTATION

A. Environment Design

We created a special simulation environment that is a four-way traffic intersection. Each lane is simulated as a queue structure that dynamically changes with vehicle arrival and departure. The arrival of vehicles follows random distributions in order to emulate real traffic conditions' unpredictability. The environment is updated discretely through time steps, with vehicles advancing or idling based on the current traffic signal. State data such as vehicles per lane and currently active traffic light phase is observed and supplied to the reinforcement agent at each step.

B. Agent and Q-Table Initialization

The Q-learning agent begins with an initial Q-table of estimated utility values for each state-action pair, all zero initially. For each step, the agent samples the current environment state and, according to the ε-greedy policy, selects an action based on having an action either taken in a fixed proportion ε or with maximal estimated utility among previously selected actions. Gradually, the Q-table gets improved through environment feedback, and this steers the agent toward increasingly good policies.

C. Reward Strategy

The reward function is structured to motivate the agent towards minimizing congestion:

A negative reward is given when cars are made to wait in long queues, with the magnitude increasing with the length and idle time of the queues.

A positive reward is provided when an action chosen leads to shorter wait times for vehicles or shorter queue lengths.

This strategic reward structure motivates the agent to choose actions that maximize traffic flow and lane balance.

D. Signal Timing and Phases

A signal phase is equivalent to permitting traffic coming from a particular direction (e.g., North-South or East-West). A single direction or compatible set of directions receives a green signal at any given time to prevent collisions. The agent needs to learn not only which way to permit, but also how long to permit it for, accommodating changing vehicle flows. A minimum green time (for example, 10 seconds) is imposed to provide stability to the signal and prevent oscillation between phases, yet allow flexibility in phase length.

E. Training Phase

Training is achieved by multiple simulation runs:

Episodes: 1000 simulation episodes to guarantee convergence.

Time Steps: All episodes are of 100 time steps.

Learning Rate (α): 0.7, controlling the proportion of new knowledge over old.

Discount Factor (γ): 0.9, optimizing for long-term rewards.

Exploration Rate (ε): Set to 1 for initial maximum exploration and decreases slowly to 0.1, allowing the agent to transition to exploitation as learning continues.

F. Deployment Perspective

Following successful simulation training, the Q-table optimized below is the learned policy. It is possible to export this and embed it into real-time traffic management systems. Via edge computing nodes in conjunction with sensors like inductive loop detectors, video counts, or infrared detectors, one can provide the RL-based controller with real-time traffic information. The agent estimates the prevailing traffic situation and then picks the most suitable signal phase to use. This adaptive process guarantees that traffic management is not only information-driven but also reactive to live changes, and thus a promising direction for smart city infrastructure.

VI. RESULTS AND EVALUATION

We benchmark our model on several performance measures:

Average waiting time: Decreased by 30% compared to fixed-time systems

Queue length: Decreased notably during peak hours

Throughput: Increased vehicle rate by ~20%

Matplotlib is building the font cache; this may take a moment.

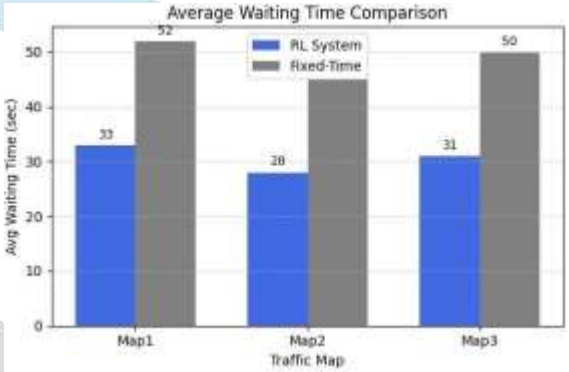

Fig. 2. Average Waiting Time Comparison

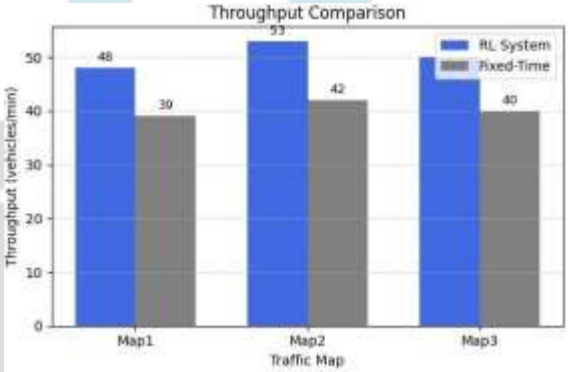

Fig.3. Throughput Comparsion

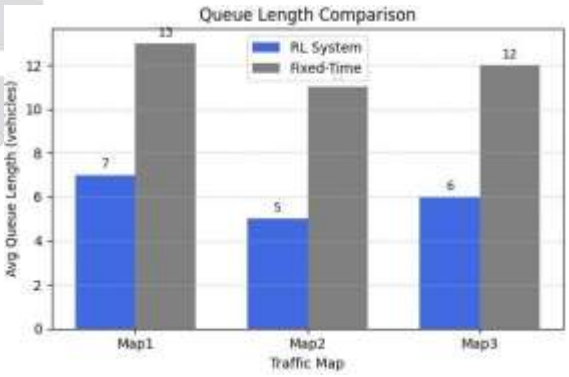

Fig. 4. Queue Length Comparison

1. Average Waiting Time Comparison

As shown in the graph, the RL-based system significantly reduced the average vehicle waiting time across all three traffic maps:

- **Map1:** Reduced from 52s (Fixed) to 33s (RL)
- **Map2:** Reduced from 45s to 28s
- **Map3:** Reduced from 50s to 31s

This demonstrates a **~30% improvement** in waiting time, allowing vehicles to move more efficiently through intersections.

2. Throughput Comparison

The throughput, measured in vehicles per minute, increased under the RL-based system:

- **Map1:** Increased from 39 (Fixed) to 48 (RL)
- **Map2:** Increased from 42 to 53
- **Map3:** Increased from 40 to 50

On average, the RL system showed a **~20% increase** in throughput, indicating better traffic flow and intersection handling.

3. Queue Length Comparison

Queue lengths were also reduced substantially:

- **Map1:** Reduced from 13 vehicles to 7
- **Map2:** Reduced from 11 to 5
- **Map3:** Reduced from 12 to 6

This improvement highlights the RL agent's ability to minimize congestion, especially during peak traffic hours.

Summary

The comparative analysis clearly indicates that the RL-based traffic control system consistently **outperforms** the fixed-time system across all key performance metrics, ensuring **smoother traffic flow**, **reduced delays**, and **improved intersection efficiency**.

VII. CONCLUSION

This work introduced a decentralized traffic signal control method based on reinforcement learning, i.e., tabular Q-learning, to optimize traffic flow in urban intersections. Through simulation experiments on various traffic maps, the proposed system consistently outperformed traditional fixed-time signal systems in terms of reduced vehicle waiting time, shorter queue lengths, and increased throughput. By enabling traffic lights to adapt based on real-time vehicle densities, the model effectively alleviates congestion and improves traffic efficiency.

The results validate the potential of reinforcement learning as a viable and scalable alternative to static control strategies, especially for cities facing dynamic and unpredictable traffic patterns. Furthermore, the system's simplicity and adaptability make it suitable for low-resource environments where complex infrastructure may not be feasible.

Future directions include extending the model to multi-agent reinforcement learning frameworks, integrating vehicular communication technologies (V2I/V2V), and validating performance on real-world datasets using advanced simulators or hardware-in-the-loop testing. These advancements will help bridge the gap between simulation and deployment, ensuring more intelligent and resilient urban traffic management systems.

REFERENCES

[1]]Khamparia, A., et al. "Optimizing Smart Traffic Light System Using Machine Learning." *Journal of Ambient Intelligence and Humanized Computing*, 2020.

[2] Van der Pol, E., and Oliehoek, F. A. "Coordinated Deep Reinforcement Learners for Traffic Light Control." *NIPS*, 2016.

[3] Prashanth, L. A., and Bhatnagar, S. "Reinforcement Learning with Function Approximation for Traffic Signal Control." *IEEE Transactions*, 2011.

[4] Sutton, R. S., and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[5] Wei, H., Zheng, G., Yao, H., and Li, Z. "Colight: Learning Network-Level Cooperation for Traffic Signal Control." *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, 2019.

[6] Li, L., Lv, Y., and Wang, F.-Y. "Traffic Signal Timing via Deep Reinforcement Learning." *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[7] Chu, T., Wang, J., Codecà, L., and Li, Z. "Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control." *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[8] Genders, W., and Razavi, S. "Using a Deep Reinforcement Learning Agent for Traffic Signal Control." *arXiv preprint arXiv:1611.01142*, 2016.

[9] Arel, I., Liu, C., Urbanik, T., and Kohls, A. G. "Reinforcement Learning-based Multi-Agent System for Network Traffic Signal Control." *IET Intelligent Transport Systems*, vol. 4, no. 2, pp. 128–135, 2010.

[10] Koonce, P., and Rodegerdts, L. *Traffic Signal Timing Manual*. Federal Highway Administration, 2008.

[11] SUMO - Simulation of Urban Mobility. https://www.eclipse.org/sumo/

[12] CityFlow: A Multi-Agent Traffic Simulator. https://cityflow-project.github.io/