SUMO-Based Intelligent Traffic Lights Control System For Congested Areas in Colombo

Master Project

In partial fulfillment for the degree of Master of Science in Information Technology

December 24, 2024

The Kyoto College of Graduate Studies for Informatics
Applied Information Technology
Web Business Technology

Chamod Kanishka Chathuranga
M23W0628

（Spring 2023, Data Science）

Title	SUMO-Based Intelligent Traffic Lights Control System For Congested Areas in Colombo		
Student ID Number	M23W0628	Name	Chamod Kanishka Chathuranga
Project Sponsor Name		Prof. Terashita	

Abstract

This research addresses the escalating issue of traffic congestion in the Colombo Metropolitan Area, Sri Lanka, where rapid urbanization and increasing transportation demand have exacerbated the strain on existing infrastructure. Even with multiple traffic management strategies like fixed-time signal control and manual efforts, congestion continues, resulting in financial losses, longer travel durations, and higher fuel usage. This research proposes a new method for traffic management that leverages reinforcement learning (RL) to regulate traffic signals in real-time dynamically. Through reinforcement learning, the proposed system aims to improve traffic movement, reduce waiting periods, and boost the overall efficiency of city transport. A method based on simulation is utilized, leveraging the open-source traffic simulation software SUMO (Simulation of Urban MObility) to illustrate traffic situations and assess the effectiveness of the proposed system. The findings suggest that applying reinforcement learning to traffic signal control can reduce congestion, improve road safety, and offer a sustainable solution to urban transportation challenges. This study contributes to the growing area of intelligent transportation systems and establishes a foundation for more flexible and responsive traffic management approaches in fast-developing urban regions.

Keyword: Traffic signal, SUMO, Traffic simulation, Traffic flow prediction, Predictive analysis, Reinforcement learning

Table of Contents

Introduction

Traffic congestion continues to be a major issue in many cities worldwide, particularly in developing nations. It leads to significant delays, excessive fuel consumption, and financial losses. Poorly designed road systems often result in small key areas within numerous developing countries that frequently become congestion hotspots. Insufficient traffic management in and around these critical areas is likely to lead to prolonged traffic jams. [1]

Sri Lanka has an extensive road network of about 119,000 kilometers, with a high road density of 1.7 kilometers per square kilometer compared to regional peers. This network includes National, Provincial, Pradeshiya Saba, and Local Authority roads and those built under development projects. The 12,380 kilometers of National roads—comprising class “A” (trunk), class “B” (main), and Expressways—are managed by the central government through the Road Development Authority (RDA). [2]

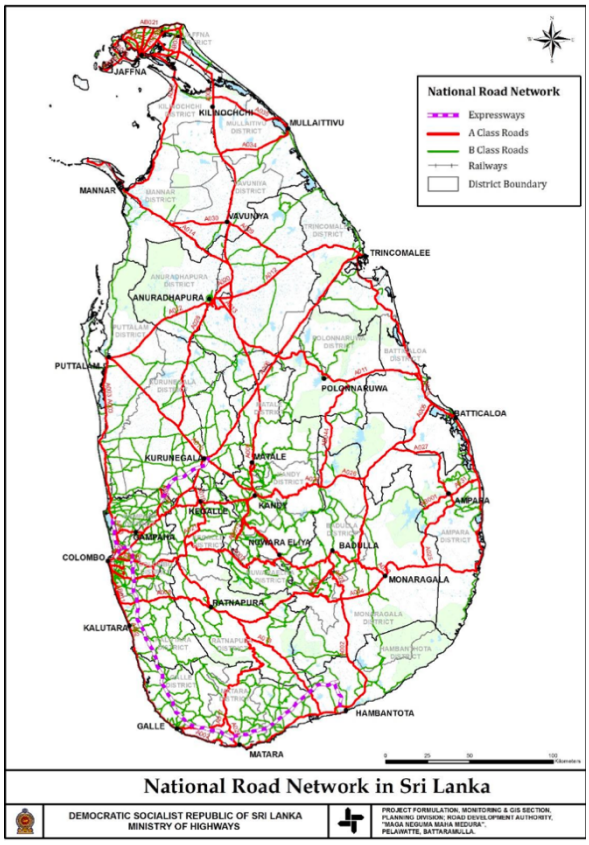


Figure 1

Transportation demand has significantly increased in recent decades, especially in the Colombo Metropolitan Area. Colombo is the economic center of Sri Lanka, with government and private offices, factories, hospitals, hotels, schools, and the harbor driving high daily travel demand. This central location also attracts many ad-hoc travelers, as it serves as the main road and rail transport hub. As a result, most travelers from the western region to other parts of the country must go through Colombo. [3]

Table 1-1: Important Buildings in Colombo City

Feature	Nos.
Police Stations	19
Primary and Secondary Schools	132
Government and Private Banks	97
Government offices	161
Government and private Hospitals	40
Embassies	31
Hotels	13
Theatres & Cinemas	27
Places of Worships	53
Bus Stoppings and Terminals (Fort and Pettah only)	10
Railway stations	6

Figure 2

(Only in Greater Colombo)

As traffic demand has increased, congestion has also escalated, resulting in several adverse effects. This includes economic losses, costs associated with travel time, and higher operating expenses for vehicles, including fuel consumption [1]. Even though various measures have been implemented to reduce traffic congestion in Colombo, the majority have not yielded positive results. This indicates that the issue is escalating rapidly and needs to be tackled continuously to prevent adverse effects. [1] The primary cause of congestion in cities is that the roads are overcrowded with vehicles exceeding their capacity. Colombo, being the central hub that offers numerous job opportunities and serves as the Commercial Capital of Sri Lanka, consistently experiences heavy traffic, particularly during peak hours. [2]

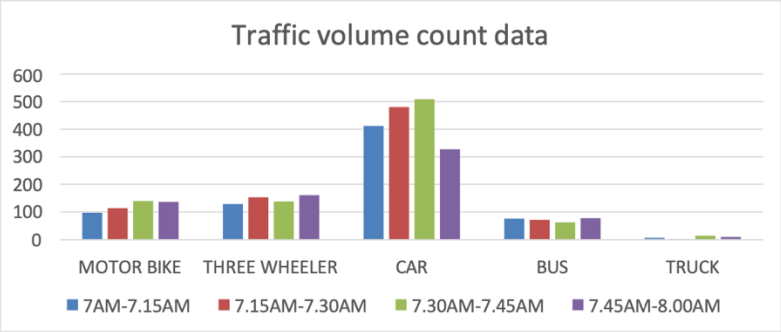


Figure 14 – Traffic Volume Count Data

Figure 3

City centers experience heavy traffic during rush hours, particularly at intersections. Significant delays occur during school drop-off times from 7:15 to 7:45 a.m., followed by commuter traffic from 8:00 to 9:00 a.m. In the afternoons and evenings, congestion is noted from 1:30 to 2:30 p.m. and again from 5:00 to 8:00 p.m. [5]. Ineffective intersection control strategies are a primary cause of traffic issues. Managing traffic at intersections during peak hours, when the highest volumes occur, is increasingly challenging [5]. Road users encounter difficulties at intersections because of extended queue lengths and longer waiting times when traffic signals are in operation. To alleviate this issue, traffic officers are deployed to manage traffic at intersections during busy periods [6]. The cause was thought to be that the signal cycle timings could not be adjusted to improve the situation for the increased traffic flow rates [6]. During peak hours in Sri Lanka, traffic police manage most city center intersections, with grade-separated intersections used in severe traffic. They primarily focus on clearing queues from major roads, giving less attention to minor roads. Thus, it's important to evaluate the effectiveness of manual traffic control versus traffic signals [5].

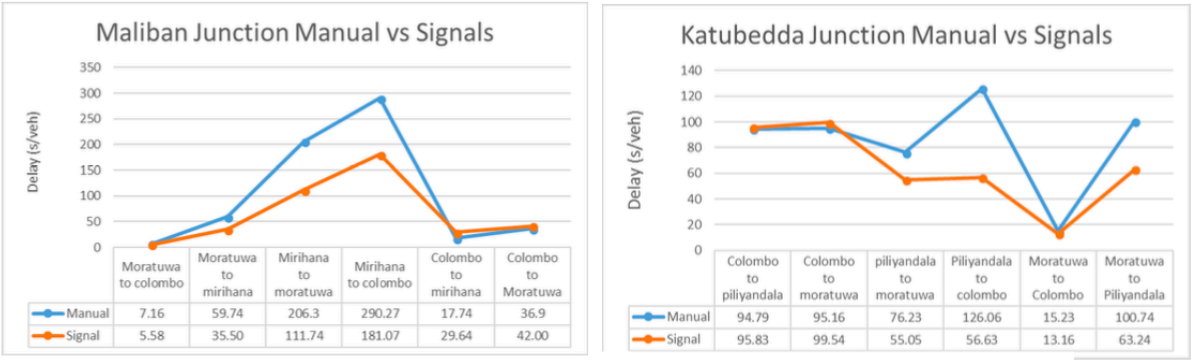


Figure 3 Vehicle delay Manual Control vs Traffic Signals

Figure 4

Research shows that fixed traffic light timings and police management are ineffective during peak hours. These methods fail to adapt to changing traffic conditions, leading to delays. Adaptive traffic control systems are needed to improve flow and reduce congestion.

Proposed Approaches

Signal Coordination Approach

This is one approach proposed by previous researchers, Traffic signal coordination involves linking multiple traffic signals to enhance the flow of specific directional movements. It sets the timing of green lights along a series of signals to ensure smooth traffic flow, especially on busy arterial roads with frequent signals [7]. Signal coordination has drawbacks, including negative community impacts, increased traffic, high maintenance costs, and the need for qualified staff.

Despite reducing stops and delays, factors like traffic growth, complex layouts, and incidents can hinder free-flow travel. [7]

Traffic Queue Length Approach

This is another approach proposed by researchers. The proposed system hardware comprises sensors, which are simple and well-known for estimating queue lengths. However, a single sensor is insufficient; therefore, an array of sensors is necessary, arranged to maximize traffic data output. The sensor topology is vital for accurate queue length estimation. [8]. Sensor numbers can vary by direction and should be sized based on peak-hour traffic. The practical placement and spacing of sensors depend on vehicle types, lengths, and traffic conditions. [8] This approach is not cost-effective, and it is difficult to install sensors on roadways.

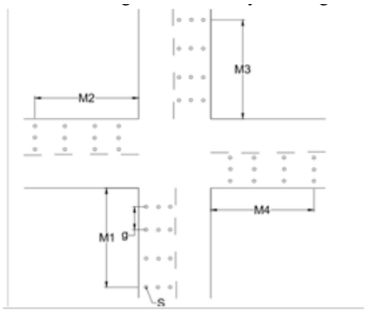


Figure 5

Introducing a New Approach

Reinforcement Learning

This study introduces reinforcement learning (RL) to overcome the limitations of fixed-time signal control systems. RL is a branch of machine learning that enables intelligent agents to adjust traffic signal timings based on real-time traffic conditions, optimizing a specified reward [9].

The goal of the RL approach is to minimize delays, reduce congestion, shorten travel durations, enhance safety, and improve traffic flow [10]. This method could enhance the coordination of traffic signal controllers, which in turn may reduce traffic congestion in urban areas. By implementing reinforcement learning, traffic light systems can become more intelligent and able to adapt to changing conditions [10]. These algorithms rely on data to use negative reinforcement to deter congestion while employing positive reinforcement to promote smooth traffic flow. As a result, urban transportation networks can evolve to be more intelligent and flexible [10]. The dynamic characteristics of this innovative traffic management strategy are illustrated in Figure 1, which outlines how the state interacts with the environment and the decision-making process involved [9].

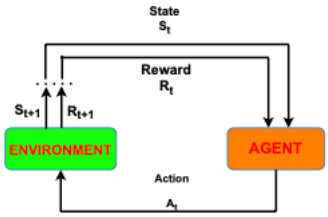


Fig. 1. Reinforcement Life Cycle

Figure 6

Simulation of Urban MObility – SUMO

In 2001, the German Aerospace Center (DLR) initiated the creation of SUMO, an open-source traffic simulation software package. [3]

SUMO is designed to replicate a traffic road network that resembles a city. Since the simulation is multi-modal, it encompasses not only the movements of cars within the urban layout but also the public transportation systems along the street network, including various train networks [4]. it features a robust simulation engine suitable for modeling everything from individual intersections to entire metropolitan areas, along with a "remote control" interface (TraCI) that allows for real-time adjustments to the simulation. [3]. SUMO represents a comprehensive set of applications designed to assist in the preparation and execution of traffic simulations rather than being just a traffic simulation itself. In order to simulate traffic using "sumo," it is necessary to represent road networks and traffic demand in a specific format, which means that both need to be either imported or created from various sources [3].
A person's identity is influenced by their departure time and the path they select, which encompasses various sub routes that dictate a particular mode of transportation. For example, in a theoretical scenario, an individual might drive their vehicle to the nearest public transport station and then continue their journey using various forms of transport. While a person may also choose to walk, this action is not simulated directly; instead, it is accounted for by calculating the time it would take for them to arrive at their destination. [4]

Figure 2: Multimodality

Traffic flow is modeled microscopically, meaning each vehicle is represented individually with specific locations and speeds. At every 1-second time step, these values are updated based on the

vehicle in front and the roadway characteristics. This simulation operates in a time-discrete and space-continuous manner, as detailed in the findings of SUMO [4].

SUMO functions as a purely microscopic traffic simulation. Each vehicle is explicitly defined, identified by its name, time of departure, and route within the network, as outlined in the research from SUMO [3]. Additional details about each vehicle, such as lane utilization, speed, and position, can also be specified if desired. Furthermore, vehicles can be categorized by type to reflect their physical characteristics and the parameters of the movement mode employed.

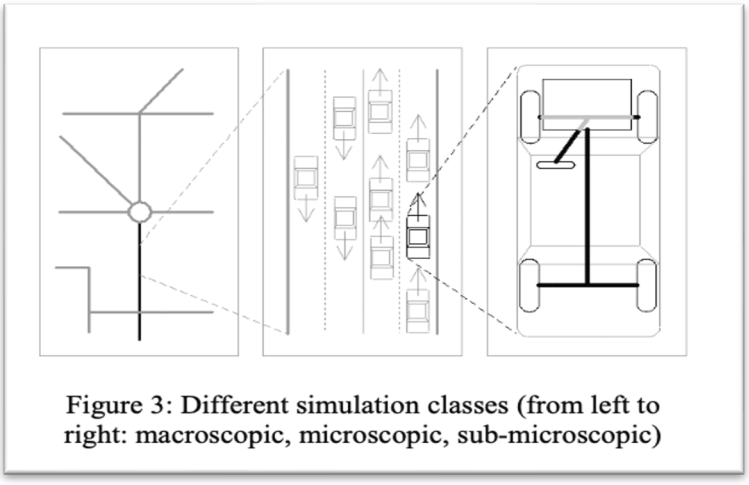


Figure 3: Different simulation classes (from left to right: macroscopic, microscopic, sub-microscopic)

Traffic signals are essential for traffic management and improving flow. Each simulated intersection can include traffic lights, and some German junctions allow right turns on red. An update to the right-of-way rules regarding this is being introduced [4].

SUMO also provides effective traffic simulation capabilities that enable vehicles to operate without collisions while supporting a range of vehicle types.. It includes multi-lane roads for lane changes and applies right-of-way rules at junctions, managing different priority scenarios. SUMO generates XML outputs that track network states over time and supports data input from multiple XML files for easier management. Detectors can also produce GnuPlot or CSV outputs [4]

Literature Review

A SUMO Based Simulation Framework for Intelligent Traffic Management System [13]

The study presents an Intelligent Traffic Management System (ITMS) incorporating a Deep-Neuro-Fuzzy model for optimizing traffic flow and alleviating congestion in metropolitan areas. Utilizing Dijkstra's algorithm, the system selects optimal routes based on road segment weights derived from the model. To assess the model's effectiveness, the authors combined it with the open-source traffic simulation tool SUMO, which allows for the simulation of various traffic-related situations, including route optimization and traffic signal control. Additionally, a custom GUI was developed for controlling simulation parameters and visualizing traffic flow feedback. The findings indicate that the model can be successfully utilized in practical scenarios, confirming its capacity to replicate realistic traffic situations and improve routing by comparing it with various pre-existing algorithms.

Simulation of modern Traffic Lights Control Systems using the open source Traffic Simulation SUMO [14]

The "OIS" (Optical Information Systems) project developed and tested innovative traffic control mechanisms using advanced sensors beyond traditional inductive loops. An agent-based traffic light algorithm was introduced, optimizing junction flow by using jam length as input. Because there were no practical testing opportunities available, the effectiveness of the algorithm was confirmed using the open-source traffic simulation software SUMO, which showed enhancements in traffic flow at intersections.

Traffic Signal Control System Based on Intelligent Transportation System and Reinforcement Learning [15]

This study proposes a Traffic Signal Control System (TSCS) leveraging an Intelligent Transportation System (ITS) architecture and a Reinforcement Learning (RL) algorithm to minimize vehicle queues at intersections. The system includes components for queue detection using computer vision and an RL-based signal program. Development stages involved system architecture design, prototype creation, and rigorous testing, including module-specific and real-time integration tests. Simulations in a medium-sized city showed significant improvements, reducing vehicle queues by 29%, waiting time by 50%, and lost time by 50% compared to fixed traffic signal phases.

Traffic Adaptive Control Framework for Real Time Large-Scale Emergency Evacuation [16]

This study presents a traffic adaptive control framework for real-time large-scale emergency evacuations, aimed at efficiently managing evacuation traffic flow. The framework combines a prescriptive reference model, which specifies desired traffic states in a rolling-horizon manner, with an adaptive control system that uses real-time traffic data from sensing systems. The adaptive control system devises and implements traffic management strategies to guide traffic towards

desired conditions, ensuring a smooth and efficient evacuation during both natural and man-made emergencies.

Traffic Sub-area Division Expert System for Urban Traffic Control [17]

This paper discusses a method for effectively controlling traffic networks by dividing them into sub-areas for better coordination and reduced complexity. It introduces an expert system based on an integrated correlation index to balance multiple demands in traffic control. The system's framework includes a knowledge representation method and a fuzzy logic-based inference engine. A case study in Zhangjigang City, China, demonstrates that the system successfully addresses multiple traffic demands, and its results align closely with those of transportation experts.

Methodology

Here are the reinforcement learning approaches and algorithms that were studied for implementation.

Q-Learning Method

Reinforcement learning is increasingly popular. It is applied in areas like game theory and operations research, allowing computers to make decisions without historical or labeled data [11] . Q-learning is a type of model-free reinforcement learning. It can also be seen as a technique for asynchronous dynamic programming (DP). This approach enables agents to learn how to act optimally in Markovian environments by experiencing the outcomes of their actions without needing to create representations of those environments. [12]

MARKOV DECISION PROCESS/ Markovian environments—An MDP is a framework for sequential decision-making. The environment is probabilistic, so the outcomes of actions are random. Policies dictate how actions are chosen in a state, and MDPs are used to create reinforcement learning algorithms. [11]

Deep Q Network Method

The term ‘Deep’ in ‘Deep Q-Networks’ (DQN) highlights the use of Deep Convolutional Neural Networks (CNNs). These CNNs are inspired by the way the human visual cortex interprets images and are crafted to analyze and retrieve features from visual information. [15] In this setting, CNNs work on unprocessed image data to recognize objects and pinpoint their locations. Following this, the improved data is sent to the agent, resulting in a clearer representation of the environment to improve decision-making abilities. [15]

Google’s Deep Mind developed the Deep Q Network (DQN) by combining Q Learning and Deep Learning. This algorithm learned to play 49 Atari games, often surpassing human high scores. [15]. We can gain insight into the strength of this algorithm. This study aims to evaluate the performance of this algorithm in managing traffic signal timing for large area maps using reinforcement learning.

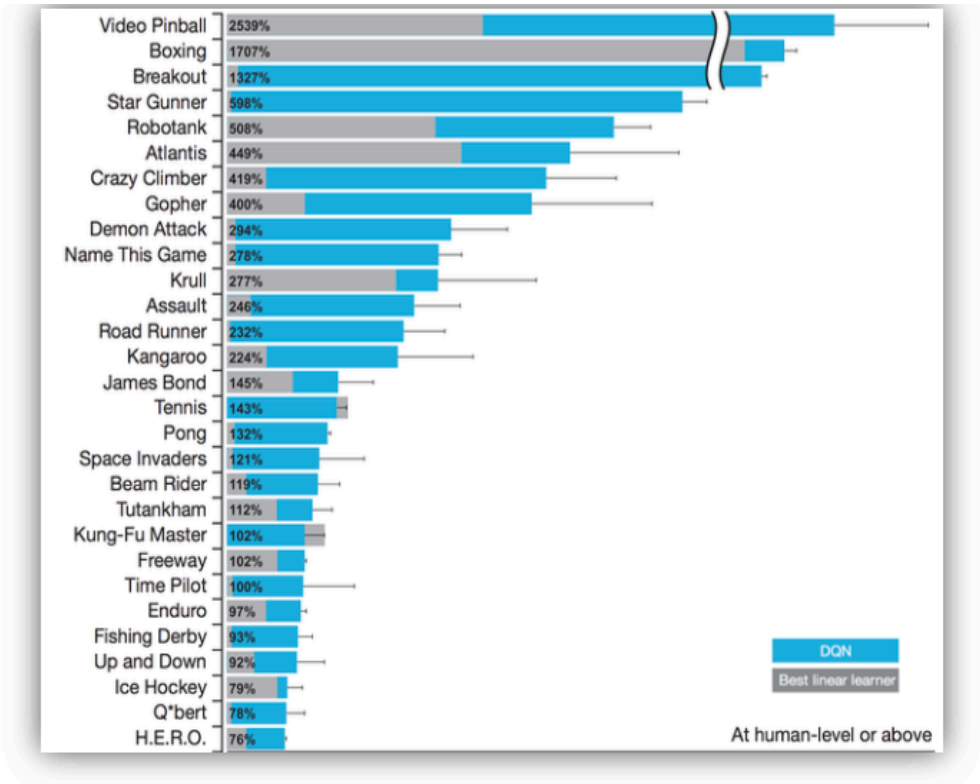


Figure 1- Normalized performance of DQN vs. human gamer (100(DQN-score – random-play-score)/(human-score-random-play-score)) for games where DQN performed better than human gamer. (Ref: DQN-Nature-Paper)*

Figure 7

Multi-Agent Reinforcement Learning (MARL)

A multi-agent system is composed of several distributed entities known as agents that independently make decisions and interact with one another in a common environment. Each agent strives to accomplish a particular goal, requiring a variety of skills to execute intelligent actions. Depending on the specific task, complex interactions between agents may arise, leading them to either work together or compete against one another to outperform their rivals [16].

MARL has been utilized across various fields, such as distributed control, telecommunications, and economics [17] . It is typically represented as either a multiagent Markov decision process (MDP) or a team Markov game in which the agents share a reward function. A more complex yet demanding cooperative scenario involves distinct reward functions exclusive to each agent, while the unified objective is to optimize the average long-term return for all agents combined [17].

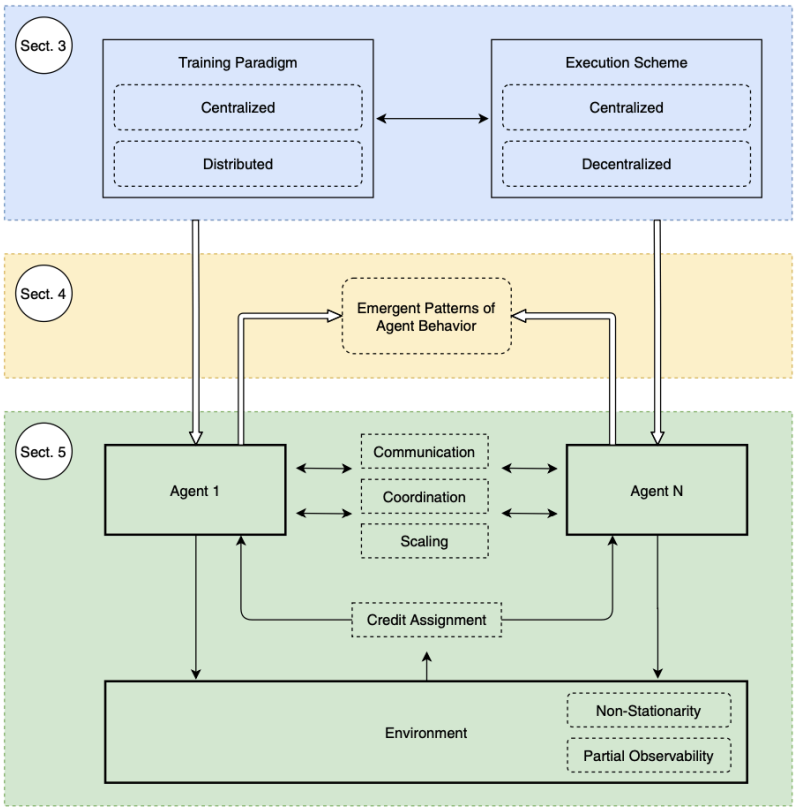


Figure 8

Decentralized MARL Network

Every agent independently decides based on the local information they observe and the communications they receive from their neighbors through the network. To optimize the globally averaged return across the network [18]. Research indicates that a decentralized method in Multi-Agent Reinforcement Learning (MARL) can be effectively used for traffic management. In this approach, each agent represents a traffic light. Each agent operates independently, attempting to optimize its performance without cooperating with the other traffic lights. Additionally, the agents are responsible for implementing their own rules for traffic management.

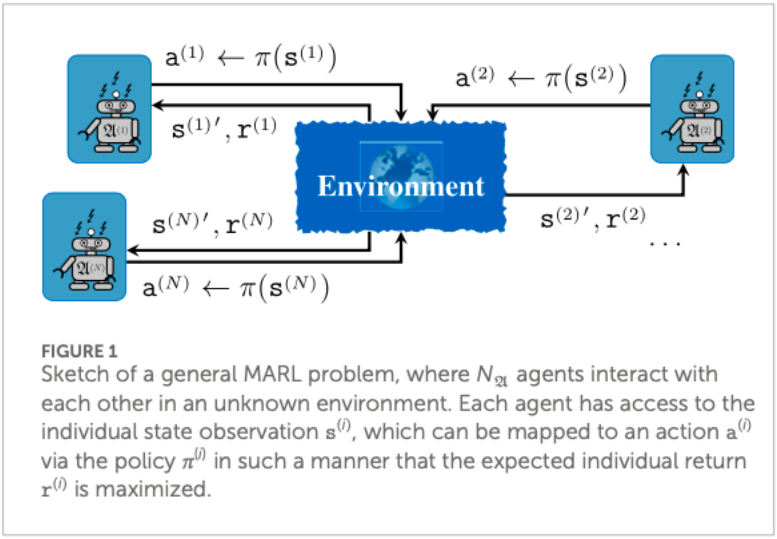


FIGURE 1
Sketch of a general MARL problem, where $N_{\mathfrak{A}}$ agents interact with each other in an unknown environment. Each agent has access to the individual state observation $\mathbf{s}^{(i)}$, which can be mapped to an action $\mathbf{a}^{(i)}$ via the policy $\pi^{(i)}$ in such a manner that the expected individual return $\mathbf{r}^{(i)}$ is maximized.

Figure 9

Centralized MARL Network

Centralized MARL (Multi-Agent Reinforcement Learning) methods train multiple agents collaboratively using shared information, addressing non-stationarity in multi-agent environments. These methods include Shared Information, which uses collective data; Value Decomposition, which divides global value functions into agent-specific contributions; and Actor-Critical, which integrates policy learning with value function optimization. [19]

Methodology Comparison Chart

	Q-Learning	Deep Q Network	MARL Centerlized Approach	MARL Decenterlised Approach
EC	No	No	Coordinate entire system	No direct coordination
Algorithm use	Bellman Equation	NL	NL	NL
Model	Q-table	DNN	DNN	DNN
Communication	No	No	High (states/ actions)	Low (status)
Action	Single Agent Action	Single Agent Action	All Agent Action	Individual Action
Robustness	Depend on Single Agent	Depend on Single Agent	Depend on Every Agent	Independent Agent
CPR	Low	Moderate	High	Moderate

EC = Environment Coordination
CPR = Computational Power Requirement
NL = Neural Network
DNN = Deep Neural Network

Implementation and Results

This chapter primarily centers on the specifics of how the proposed method is implemented. It uses Reinforcement learning methods like Q-learning, Deep-Q Network, and MARL approaches to find the best one for a signal light time adjustment system for congested areas in Colombo. Implementation and Evaluation were done using an Apple Macintosh M2 Chip, 8-core CPU, 10-core GPU, 16 Core Neural Engine, and 16GB RAM.

For effective implementation, leverage Python libraries like Keras and TensorFlow—using the Metal backend for optimized performance on Apple devices. Incorporating TraCI will enhance the project with robust traffic simulation and control for impactful results.

Tools and Applications used

SUMO GUI

Figure 10

This application is mainly designed to carry out simulations that pinpoint traffic congestion bottlenecks. It accurately replicates real-world conditions by including diverse vehicle categories—such as cars, buses, motorcycles, pedestrians, and trucks—that are relevant to the Colombo region.

I selected the Borella Junction area in Colombo for my simulation due to its unique urban dynamics and complex traffic patterns, which are likely to provide valuable insights for my study.

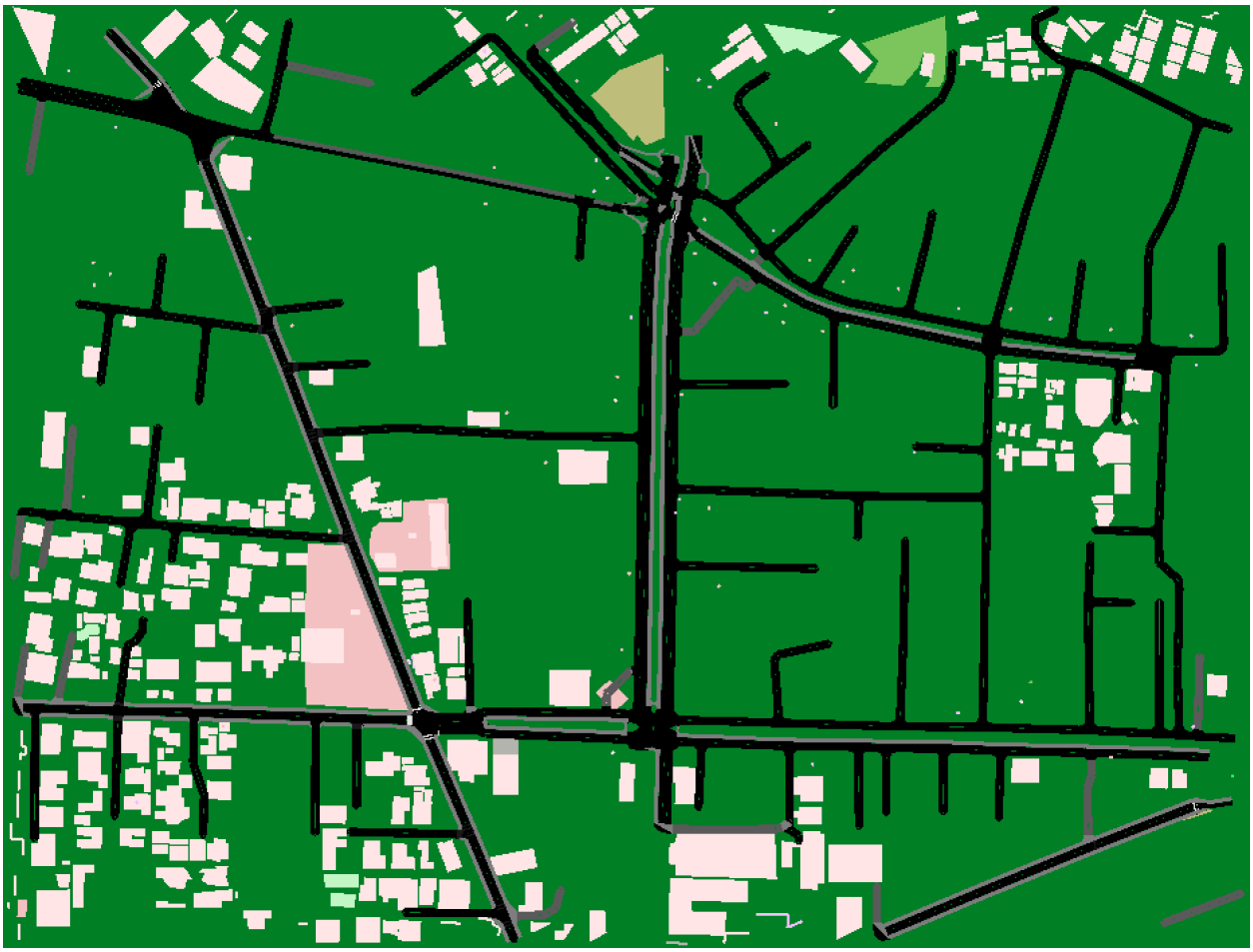


Figure 11

The image illustrates the Borella area in a sumo simulation, highlighting urban dynamics and interactions among people and vehicles. This powerful tool provides essential insights for informed urban planning and development, improving Borella's overall functionality and livability.

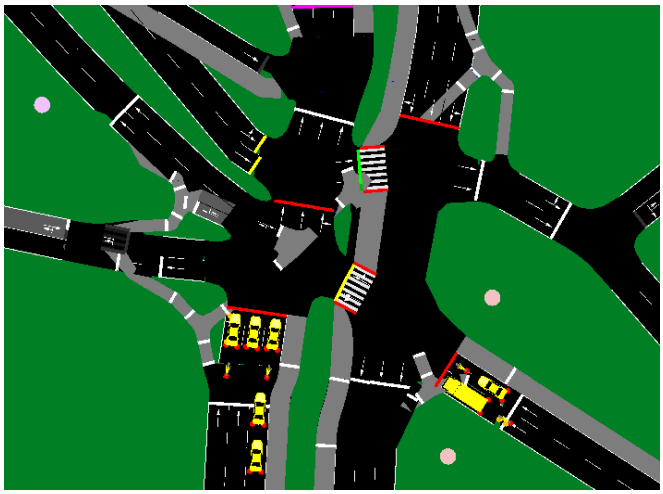


Figure 12

This image shows the main junction in the Borella area. It illustrates the complexity of this junction.

Netedit

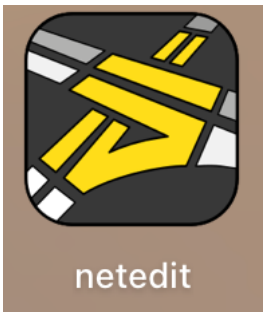


Figure 13

The primary objective of this application is to facilitate the management and modification of maps in accordance with specific rules and conditions set by the user. It offers a diverse array of tools that enhance usability and efficiency, thereby streamlining the mapping process.

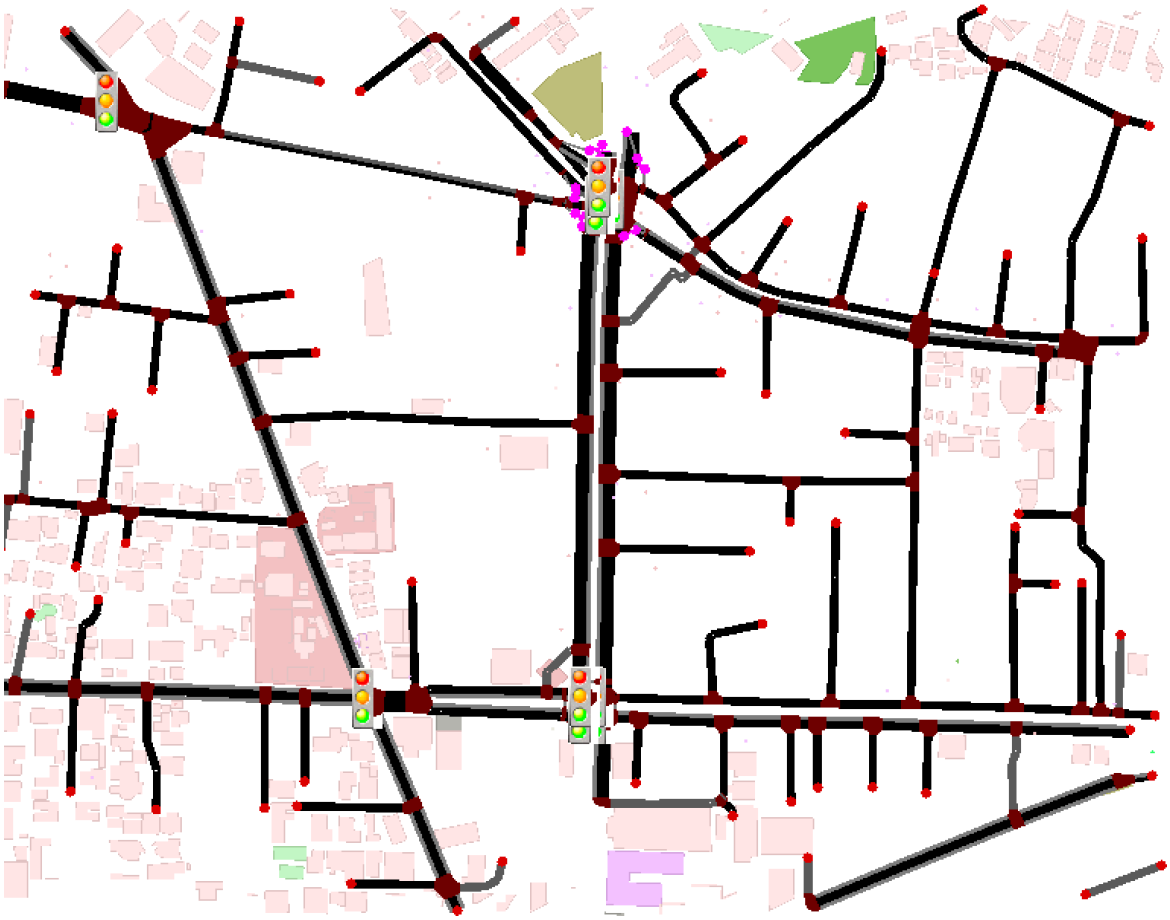


Figure 14

The image illustrates the Borella area on the Netedit map, highlighting four main junctions with traffic lights. We aim to optimize the timing of these lights using reinforcement learning to improve traffic flow and reduce congestion based on real-time conditions.

OSM Web Wizard

Figure 15

The OSM Web Wizard is a useful application for creating realistic maps for simulation. It provides various tools for generating different vehicle types and integrates with OpenStreetMap to ensure accurate map creation.

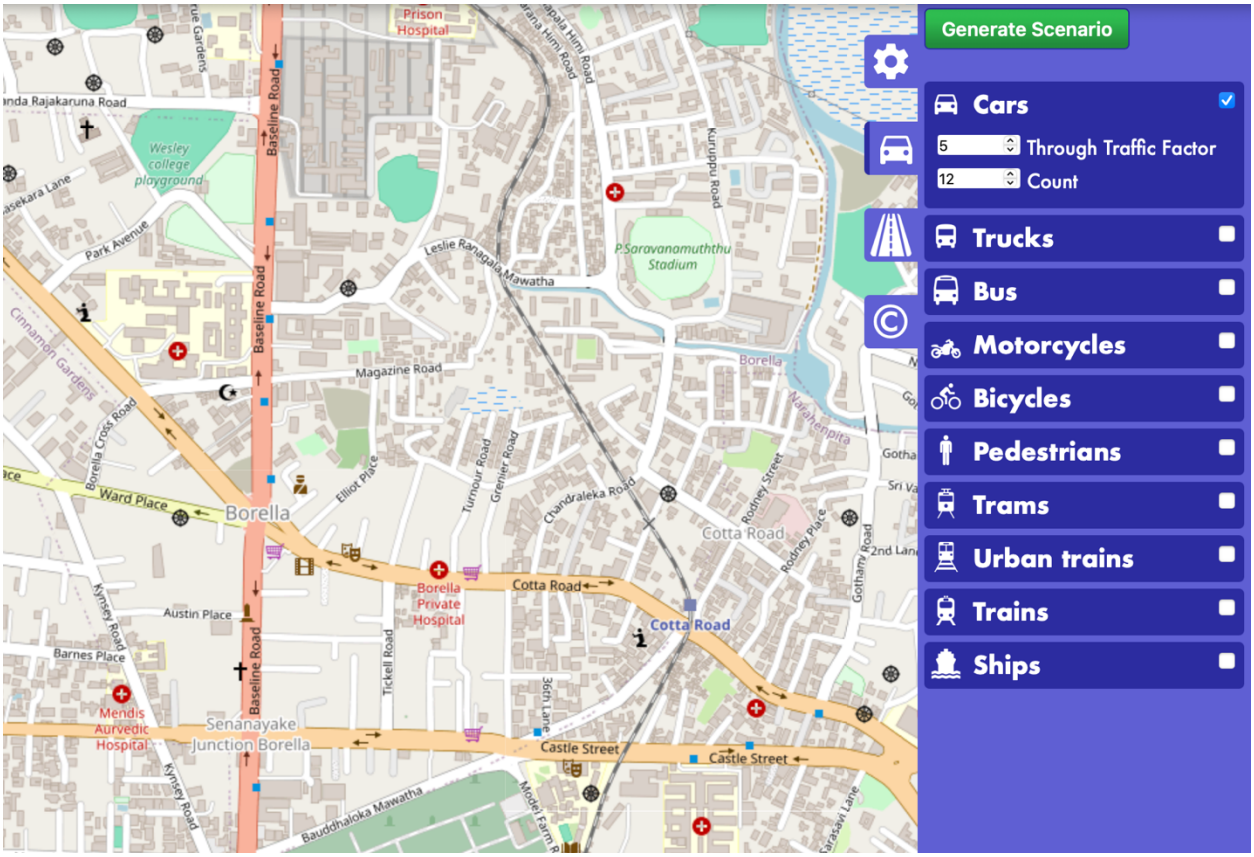


Figure 16

The image presented illustrates the OpenStreetMap (OSM) Web Wizard, a tool designed to facilitate the creation of realistic and user-friendly maps. This innovative platform leverages collaborative mapping techniques and open-source data to enhance the accuracy and detail of geographic representations. By simplifying the mapping process, the OSM Web Wizard empowers users to contribute effectively to the growing body of geographic information, fostering greater accessibility and engagement within the field of cartography.

Implementation of Proposed method

Q-learning Approach implementation

In this implementation, the agents represent Traffic Lights within the SUMO framework, following the Q-learning methodology. When we use a map as input, each map contains several traffic lights. Each traffic light has a distinct ID and unique characteristics. Based on this strategy, each traffic light operates as an individual agent.

Pseudo Code Implementation

Setup Enviroment,varibles and q-learning parameters

```
IMPORT necessary libraries (numpy, traci)
DEFINE traffic lights and their phases
INITIALIZE Q-table for each traffic light
SET Q-learning parameters (learning rate, discount factor,
exploration rate, episodes)
```

Decides next actions using epsilon-greedy policy

```
DEFINE choose_action(state, traffic_light)
IF random value < exploration rate
RETURN random action (exploration)
ELSE
RETURN action with the highest Q-value (exploitation)
```

Move simulation forward

```
DEFINE simulate_step()
    ADVANCE the SUMO simulation by one step
```

Training Q-learning model with the simulation

```
DEFINE train_q_learning()
    FOR each training episode
        START SUMO simulation with configuration file
        WHILE simulation is not finished
            FOR each traffic light
                GET current state of the traffic light
                SELECT an action using choose_action()
                EXECUTE the action (set traffic light phase)
                ADVANCE the simulation by one step
                CALCULATE reward based on traffic conditions
                UPDATE Q-table using the Q-learning formula
        CLOSE SUMO simulation
```

Evaluate the Q-learning model after trained

```
DEFINE test_q_learning()
    START SUMO simulation with configuration file
    WHILE simulation is not finished
        FOR each traffic light
            GET current state of the traffic light
            SELECT the best action based on Q-values
            EXECUTE the action (set traffic light phase)
            ADVANCE the simulation by one step
    CLOSE SUMO simulation
```

Manage input of runs train or test

```
DEFINE main()
      PROMPT user for mode (‘train’ or ‘test’)
      IF mode is ‘train’
            CALL train_q_learning()
      ELSE IF mode is ‘test’
            CALL test_q_learning()
      ELSE
            PRINT invalid mode message
```

DQN (Deep Q Network) Approach implementation

Initialize the parameters

```
Define traffic lights and their phases.
Set parameters:
- state_size, action_size, gamma, alpha, epsilon,
epsilon_min, epsilon_decay.
- batch_size, num_episodes, memory_size.
Build replay memory.
```

Configure the hyperparameters, environment, and memory storage for training.

Build DQN Agent model

```
Create a neural network with:
  - Input layer: size = state_size.
  - Two hidden layers: 32 neurons each, ReLU activation.
  - Output layer: size = action_size, linear activation.
Compile model with Adam optimizer and MSE loss.
```

Create neural network to work as DQN agent

Define core functions

```
If random value ≤ epsilon:
    Select a random valid action for the traffic light.
Else:
    Predict Q-values using the model.
    Select action with max Q-value from valid actions.
```

Execute the epsilon-greedy strategy.

Replay Function

```
If memory size < batch size, return.
Sample minibatch from memory.
For each sample:
  Compute target:
    If not done: target = reward + gamma × max Q-value of next
state.
    Update Q-value for the action in the sample.
Train the model on updated Q-values.
Decay epsilon.
```

Using previous experiences to train the model.

Process State

```
Get current traffic light phase.
Encode phase as one-hot vector.
Return vector reshaped for input to the model.
```

Transform the state of the traffic light into a format that can be utilized by the DQN model.

Train Model

```
For each episode:
  Start SUMO simulation.
  While simulation is running:
    For each traffic light:
      Get current state.
      Choose an action using Choose Action.
      Set traffic light phase.
      Step simulation forward.
      Compute reward and get next state.
      Store (state, action, reward, next_state, done) in
memory.
      Perform Replay to train the model.
  Close simulation after each episode
```

Utilize SUMO for traffic simulation and train the DQN model with the experiences gathered.

Test Model

```
Start SUMO simulation.
While simulation is running:
  For each traffic light:
    Get current state.
    Predict best action using the trained model.
    Set traffic light phase based on the action.
    Step simulation forward.
Close simulation.
```

Evaluate the trained model to confirm its effectiveness in managing traffic signals.

Main Function

```
Prompt user for mode ("train" or "test").
If mode is "train", call Train the Model.
If mode is "test", call Test the Model.
Else, show error message.
```

A primary access point that runs training or testing depending on user input.

MARL (Multi-Agent Reinforcement Learning) -Decentralized Approach implementation

Import and Initialization

```
- Import necessary libraries (numpy, traci, tensorflow, random, deque).
- Define traffic lights and their valid phases.
- Set parameters for DQN (state size, action size, gamma, alpha, epsilon, etc.).
- Create models and replay memories for each traffic light.
```

This part establishes the environment, the traffic lights, and the parameters for the DQN.

Action Selection

```
- Define a function `choose_action`:
    - If random value < epsilon, select a random action (exploration).
    - Otherwise, predict Q-values and select the action with the highest Q-value (exploitation).
```

Applies the epsilon-greedy approach to balance exploration and exploitation.

Experience Replay

```
- Define a function `replay`:
    - If replay memory size < batch size, return.
    - Randomly sample minibatches from memory.
    - Update Q-values and train the model using the loss function.
```

Employs experience replay to effectively train the model.

State Preprocessing

```
- Define a function `preprocess_state`:
    - Get the current phase of the traffic light.
    - Encode the state as a one-hot vector.
```

Transforms the unprocessed traffic light status into a format appropriate for the DQN.

Training Process

```
- Define a function `train_dqn`:
    - For each episode:
        - Start SUMO simulation.
        - Loop until simulation ends:
            - For each traffic light:
                - Preprocess the current state.
                - Choose an action using the model.
                - Apply the action and step the simulation.
                - Calculate the reward and next state.
                - Store experience in memory.
                - Perform replay to train the model.
        - Decay epsilon to reduce exploration over time.
        - Close SUMO after each episode.
```

Trains the Deep Q-Network (DQN) for every traffic signal across several episodes.

Testing Process

```
- Define a function `test_dqn`:
    - Start SUMO simulation.
    - Loop until simulation ends:
        - For each traffic light:
            - Preprocess the current state.
            - Predict the best action using the trained
model.
            - Apply the action and step the simulation.
    - Close SUMO after testing.
```

Assesses the trained models within the simulation environment.

Main Function

```
- Define a function `main`:
    - Get user input for mode (train or test).
    - Call `train_dqn` or `test_dqn` based on the input.
```

The initial point for the application, enable the user to select between training and testing modes.

(Multi-Agent Reinforcement Learning) - Centralized Approach implementation

Import and Initialization

```
- Import libraries:
    - numpy for numerical computations
    - traci for SUMO traffic simulation interface
    - random for randomization
    - collections.deque for replay memory
    - tensorflow for building neural networks
- Define traffic lights and their phases.
- Set MARL parameters:
    - State size, action size, gamma, alpha, epsilon, decay
factors, and replay memory size.
```

Prepares essential tools, configures traffic lights, and initializes the DQN model.

Action Selection

```
- Define `choose_actions(states)`:
    - If random value < epsilon:
        - Select random valid actions for each traffic light.
    - Else:
        - Predict Q-values for all traffic lights.
        - For each traffic light:
            - Select the action with the highest Q-value from
valid actions.
```

Executes a strategy that balances exploration and exploitation by using an epsilon-greedy method for action selection.

Replay Memory and Training

```
- Define `replay()`:
    - If memory size < batch size, return.
    - Sample a minibatch from memory.
    - For each experience in the batch:
        - Calculate the target Q-value using the Bellman
equation.
        - Update the predicted Q-values for the specific
actions taken.
        - Train the DQN model on the updated Q-values.
    - Decay epsilon for less exploration over time.
```

Utilizes replay memory to train the model and enhance learning stability.

State Preprocessing

```
- Define `preprocess_states()`:
    - For each traffic light:
        - Get the current phase.
        - Encode the phase as a one-hot vector.
    - Combine all one-hot vectors into a single state
representation.
    - Reshape the state for compatibility with the DQN model.
```

Converts the present traffic light statuses into a well-organized input for the model.

Training Process

```
- Define `train_marl()`:
    - For each episode:
        - Start SUMO simulation with the configuration file.
        - Loop until simulation ends:
            - Preprocess the current state.
            - Choose actions using the model.
            - Apply actions to the respective traffic lights.
            - Step the simulation forward.
            - Calculate rewards based on performance metrics.
            - Get the next state.
            - Store the experience in memory.
            - Call `replay()` to train the model.
        - Close SUMO after the episode.
```

Conducts training of the centralized MARL model across various episodes.

Testing Process

```
- Define `test_marl()`:
    - Start SUMO simulation with the configuration file.
    - Loop until simulation ends:
        - Preprocess the current state.
        - Predict the best actions for all traffic lights.
        - Apply actions to the respective traffic lights.
        - Step the simulation forward.
    - Close SUMO after testing.
    - Print "Testing completed."
```

Evaluates the trained MARL model within the SUMO environment.

Main Function

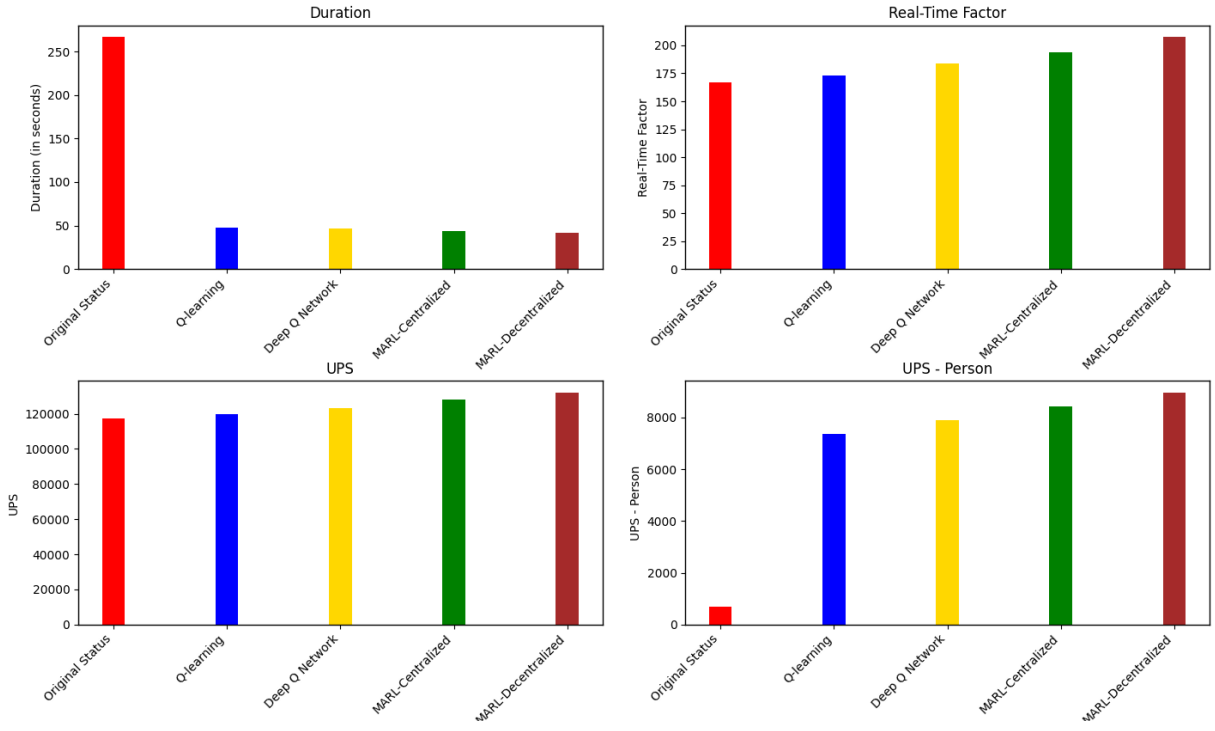
```
- Define `main()`:
    - Get user input for mode (`train` or `test`).
    - Call `train_marl()` if mode is "train."
    - Call `test_marl()` if mode is "test."
    - Print an error message for invalid input.
```

Acts as the initial access point for the program.

Results

Performance

Factor	Simulation (Original Status)	Q-learning	Deep Q Network	MARL – Centralized	MARL – Decentralized
Duration	266.77	47.64	46.78	43.89	41.32
Real-Time Factor	167.012	172.935	183.492	194.128	207.645
UPS	117244	119586	123438	127902	132154
UPS - Person	684	7360	7895	8412	8965

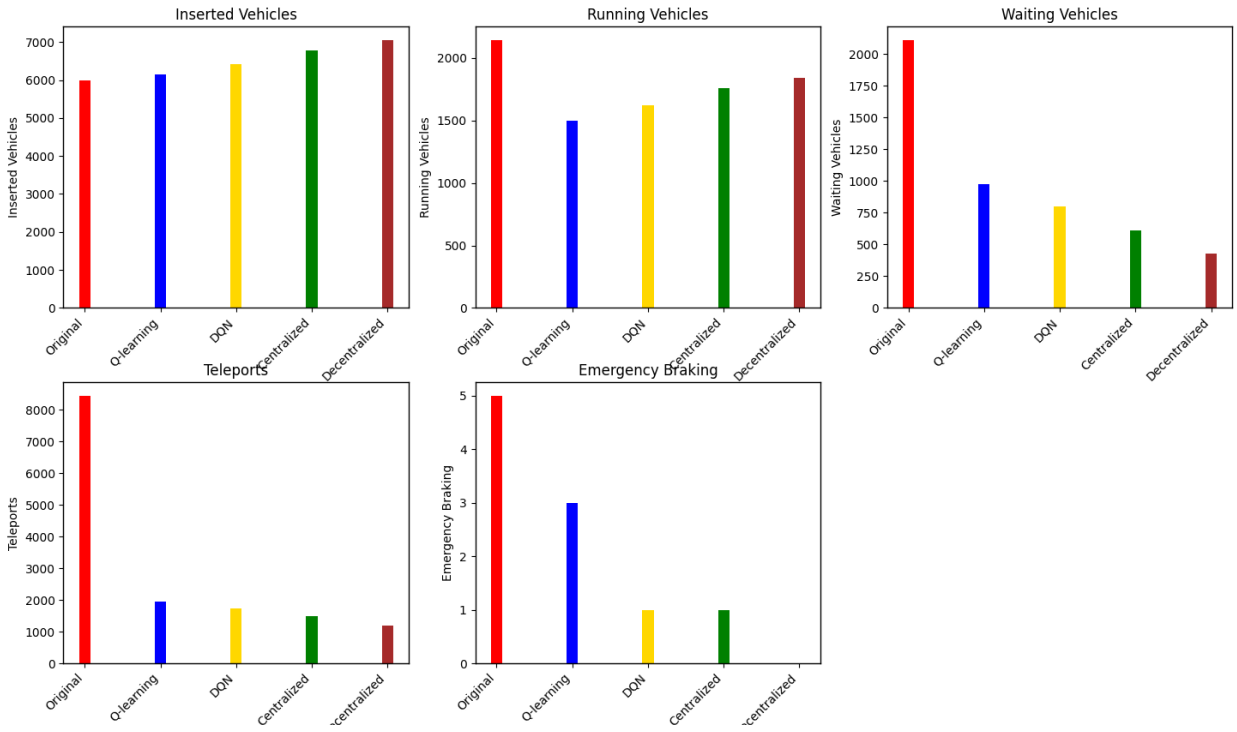


Analysis of Preformance

The Original Status model had the longest simulation time at 266.77 seconds, while machine learning models significantly reduced this duration: Q-learning (47.64 seconds), Deep Q Network (46.78 seconds), MARL-Centralized (43.89 seconds), and MARL-Decentralized (41.32 seconds), highlighting the efficiency of reinforcement learning techniques. The Real-Time Factor (RTF) increased with these models, with the Original Status at 167.012 and MARL-Decentralized at 207.645, indicating higher computational demands despite shorter simulation times. In terms of Units Per Second (UPS), the Original Status recorded 117,244, whereas MARL-Decentralized achieved 132,154, demonstrating its ability to process more units. Similarly, the Original Status for UPS-Person had a recorded value of 684, whereas MARL-Decentralized reached 8965, demonstrating improved efficiency. In conclusion, MARL-Decentralized outperforms the Original Status in performance metrics, confirming its position as the most effective solution for large-scale, real-time applications.

Vehicles

Factor	Simulation (Original Status)	Q-learning	Deep Q Network	MARL – Centralized	MARL – Decentralized
Inserted	5992	6148	6419	6792	7051
Running	2143	1497	1624	1756	1839
Waiting	2108	974	798	612	428
Teleports	8449	1943	1732	1484	1189
Emergency Braking	5	3	1	1	0

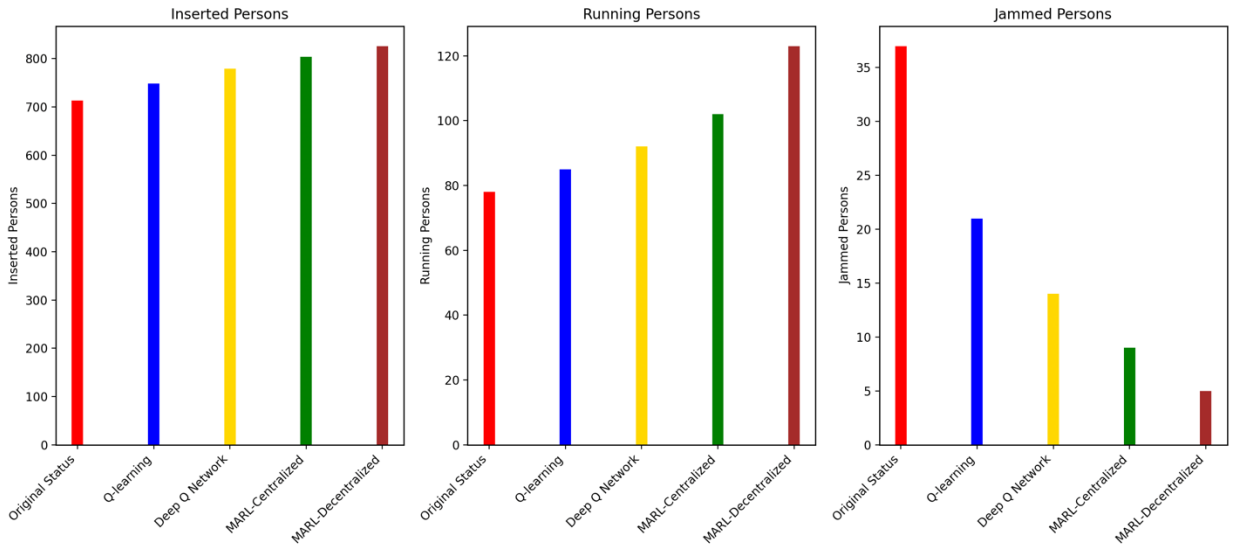


Analysis of Vehicles

The number of inserted vehicles increased from the Original Status (5992) to MARL-Decentralized (7051), showing that reinforcement learning, especially MARL-Decentralized, enhanced vehicle management and traffic flow. The count of running vehicles decreased significantly, dropping from Original Status (2143) to MARL-Decentralized (1839), indicating effective optimization in vehicle movement and reduced congestion. Waiting vehicles fell dramatically from 2108 to 428 in MARL-Decentralized, reflecting better traffic management and shorter wait times.The number of teleports, which signify vehicles circumventing the simulation, decreased from 8449 in the Original Status to 1189 in MARL-Decentralized, indicating better management. Instances of emergency braking fell from 5 to 0 in the MARL-Decentralized model, showing improved traffic flow and safety. Overall, these findings demonstrate that machine learning models, particularly MARL-Decentralized, significantly improve traffic flow, reduce congestion, and enhance system efficiency.

Persons

Factor	Simulation (Original Status)	Q-learning	Deep Q Network	MARL – Centralized	MARL – Decentralized
Inserted	713	748	779	804	826
Running	78	85	92	102	123
Jammed	37	21	14	9	5

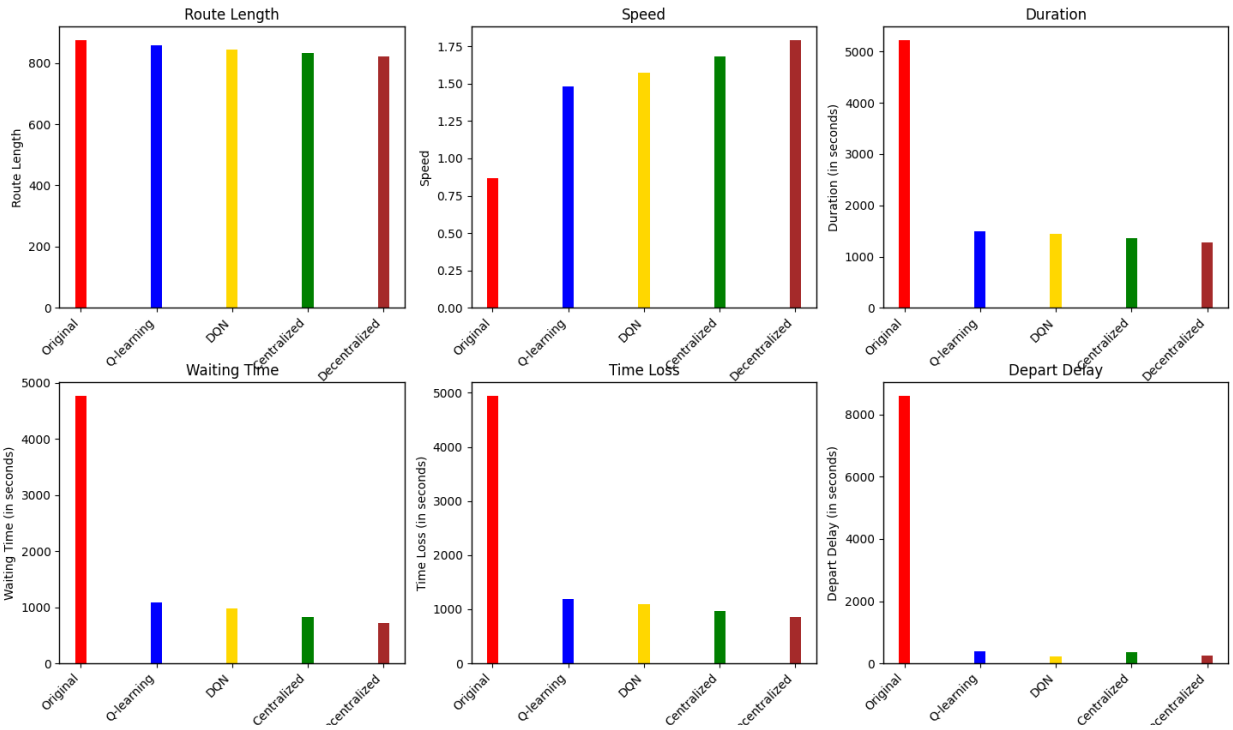


Analysis of Persons

The analysis of results highlights significant improvements in pedestrian management. The number of Inserted persons increased from 713 in Original Status to 826 in the MARL-Decentralized model, indicating enhanced management and insertion of individuals. In the same way, the number of running individuals increased from 78 to 123, indicating improved mobility and fewer delays.In contrast, Jammed persons dropped significantly from 37 to 5, demonstrating the model's effectiveness in alleviating congestion and ensuring smoother movement. In general, the results indicate that the MARL-Decentralized model significantly enhanced pedestrian movement, minimized congestion, and improved system efficiency, establishing it as the most successful model for managing pedestrian dynamics.

Statistics

Factor	Simulation (Original Status)	Q-learning	Deep Q Network	MARL – Centralized	MARL – Decentralized
Route Length	874.35	858.21	843.89	831.47	820.54
Speed	0.87	1.48	1.57	1.68	1.79
Duration	5219.81	1498.73	1439.12	1352.65	1271.98
Waiting Time	4773.45	1085.47	974.23	832.91	721.35
Time Loss	4946.44	1182.78	1093.63	973.51	862.17
Depart Delay	8607.49	389.25	223.67	348.71	256.93

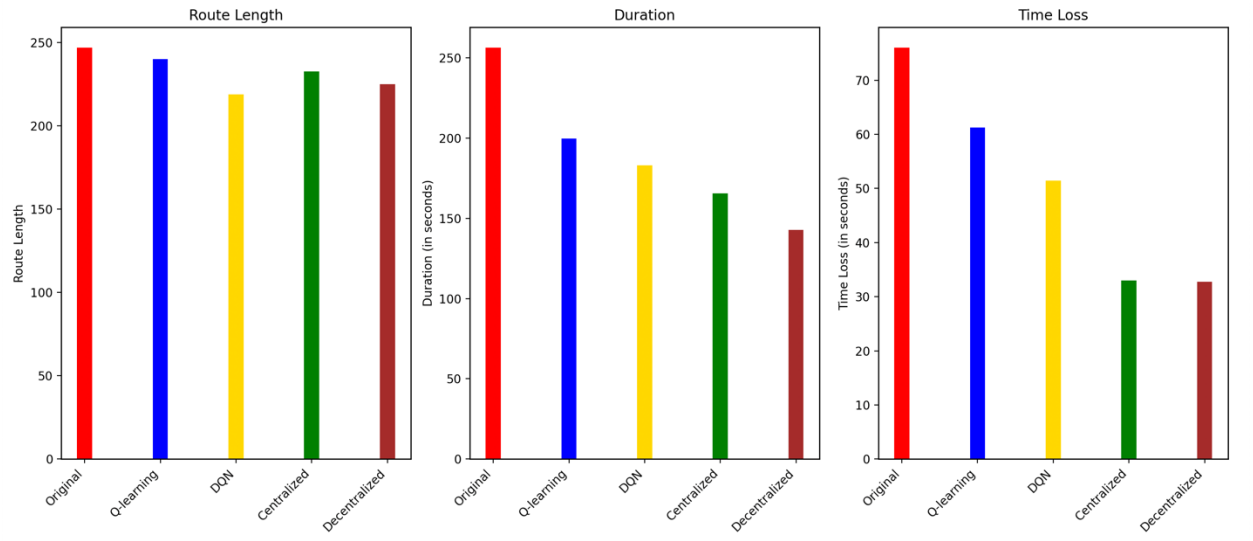

Route Length
Route Length
0
200
400
600
800
Original
Q-learning
DQN
Centralized
Decentralized
Speed
Speed
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Original
Q-learning
DQN
Centralized
Decentralized
Duration
Duration (in seconds)
0
1000
2000
3000
4000
5000
Original
Q-learning
DQN
Centralized
Decentralized
Waiting Time
Waiting Time (in seconds)
0
1000
2000
3000
4000
5000
Original
Q-learning
DQN
Centralized
Time Loss
Time Loss (in seconds)
0
1000
2000
3000
4000
5000
Original
Q-learning
DQN
Centralized
Depart Delay
Depart Delay (in seconds)
0
2000
4000
6000
8000
Original
Q-learning
DQN
Centralized

Analysis of Statistics

The analysis showed that the Route Length reduced from 874.35 in the Original Status to 820.54 in the MARL-Decentralized model, signifying better pathfinding by the machine learning models. Speed saw a notable increase, jumping from 0.87 to 1.79, which indicates improved traffic flow and vehicle efficiency. Duration was cut from 5219.81 to 1271.98, showcasing better route planning and reduced travel time. Waiting Time dropped from 4773.45 to 721.35, highlighting the models' effectiveness in minimizing idle time. Time Loss decreased from 4946.44 to 862.17, demonstrating success in reducing delays, while Depart Delay fell dramatically from 8607.49 to 256.93, emphasizing timely vehicle departures. Overall, the results demonstrate that the MARL-Decentralized model significantly enhances traffic system performance, reducing congestion and improving overall efficiency.

Pedestrian Statistics

Factor	Simulation (Original Status)	Q-learning	Deep Q Network	MARL – Centralized	MARL – Decentralized
Route Length	246.83	239.87	218.73	232.51	224.98
Duration	256.23	199.65	182.94	165.43	142.67
Time Loss	76.01	61.25	51.47	32.98	32.78



Analysis of Pedestrian

The analysis of results shows that Route Length decreased from 246.83 in the Original Status to 218.73 in the Deep Q Network (DQN) model, before rising to 224.98 in the MARL-Decentralized model, indicating better pedestrian path optimization, with DQN achieving the most improvement. Pedestrian Duration dropped significantly from 256.23 in the Original Status to 142.67 in the MARL-Decentralized model, reflecting the models' effectiveness in reducing crossing times, particularly with MARL-Decentralized outperforming the others. Additionally, Time Loss was reduced from 76.01 in the Original Status to 32.78 in the MARL-Decentralized model, highlighting the ability of these models to minimize pedestrian delays, once again with MARL-Decentralized yielding the best results. Overall, the analysis reveals that reinforcement learning models have significantly improved pedestrian movement management by shortening Route Length, minimizing duration, and decreasing Time Loss. Among the various models, MARL-Decentralized and Deep Q Network demonstrated superior performance, enhancing pedestrian movement and reducing traffic delays. This showcases the potential of machine learning to optimize pedestrian flow alongside vehicle traffic.

Discussion and Limitations

Discussion

In this project, created a reinforcement learning model aimed at controlling traffic lights, which surpasses conventional fixed-time systems by adjusting to real-time traffic conditions. Implementing this model in Colombo could significantly advantage more than a million commuters by improving traffic flow and reducing delays, allowing them to utilize their time more effectively.

This approach may also alleviate traffic congestion, resulting in lower oil consumption and diminished emissions. By optimizing traffic light timings, we can improve the air quality in the city. Ultimately, this initiative has the potential to improve residents' quality of life while fostering a more sustainable urban setting.

Limitations

Simulated Data vs Real-time Data - The research primarily relies on simulated traffic data generated by SUMO. Although SUMO offers a versatile simulation environment, the data it generates might not accurately capture the intricacies of real traffic situations in Colombo, including driver behavior, road conditions, or unforeseen events.

Lack of real-world data – The absence of systematically gathered data in real-time creates obstacles for the effective application of these strategies in practical situations. This gap restricts our capacity to evaluate and modify strategies, which impacts the success of our interventions. Tackling this problem is crucial for effective implementation.

Lack of Devices to Collect Data—The study faced challenges due to the absence of sophisticated traffic monitoring and data collection tools in Colombo. This constraint limited access to real-time and detailed traffic data, which could have enhanced the model's accuracy and practical usefulness.

Computational Complexity – Implementing and training advanced models necessitates robust computational systems to effectively simulate maps and incorporate real-world data.Robust computing resources are essential for handling complex algorithms and large datasets, ensuring the success of these applications.

Coding Experience— Building the Q-learning model and incorporating it with SUMO requires a significant level of programming expertise. Insufficient experience in crafting and executing such intricate models may result in inefficiencies, errors, and less-than-ideal outcomes throughout the development and testing stages.

Programming Research and Development - The effort emphasizes the necessity for continuous exploration in programming techniques and algorithms.Improving traffic light control with reinforcement learning requires exploring innovative reward functions, better state representation, and effective debugging, all of which necessitate focused R&D efforts.

Conclusion and Future Work

Conclusion

This research conducted a comprehensive examination of how reinforcement learning methods can be effectively applied to traffic light control. It highlighted the advantages of utilizing these sophisticated algorithms over conventional fixed-time traffic signal systems, which frequently face challenges in adapting to real-time traffic conditions.Throughout this research, I developed a deep understanding of various reinforcement learning approaches, such as Q-learning, policy gradients, and deep reinforcement learning. The unique characteristics and benefits of each approach were carefully analyzed, highlighting how these methods are influenced by core ideas such as exploration versus exploitation, reward systems, and state representation. This insight underscored the capacity of reinforcement learning to transform traffic flow management by fine-tuning signal timings, alleviating congestion, and ultimately enhancing urban mobility.

Future Work

Investigate and analyze various algorithms to determine which one is most appropriate for the specific requirements and constraints of this particular scenario. Consider factors such as efficiency, accuracy, scalability, and ease of implementation to ensure the selected algorithm effectively meets the needs of the problem at hand.

Design a device to collect real-time data. The plan is to create a device that uses video streams to gather information, such as vehicle counts and other relevant data. We also intend to install these devices at locations with traffic lights.

Focuses on preparing the models for retraining by carefully selecting the most effective parameters and methodologies. The goal is to improve their performance to the highest achievable level, making certain they can produce excellent results in different situations.

Utilize advanced high-performance computing systems to train the models, allowing for an in-depth comparison of their capabilities and efficiency.

We are preparing to evaluate the model's effectiveness by simulating actual traffic scenarios in various locations throughout Colombo city.This strategy will enable us to evaluate how effectively the model responds to the intricacies and fluctuations of urban traffic patterns, offering important insights into its practical application effectiveness.

Acknowledgment

I want to express my sincere gratitude to everyone who supported me in my research on applying a Reinforcement Learning Approach to Intelligent traffic light control.

First, I thank my supervisor, Ms. Sameera Abar, for her invaluable guidance and feedback, which greatly shaped my study. I also appreciate Professor Terashita's support and mentorship, which were crucial in motivating me to achieve my goals.

I owe much to the Kyoto College of Graduate Studies of Informatics for the resources and environment that facilitated my research. Additionally, the foundational work of previous researchers has been a constant source of inspiration.

Finally, thank you to my family and friends for their unwavering support during this challenging journey. These incredible individuals and institutions made this research possible.

Bibliography

[1] Dammulla.R, Mudunkotuwa.R, "Analysis on the Road Traffic Congestion in Colombo Metropolitan Area," *CINEC Academic Journal,* vol. 5, no. 1, p. 15, 2021.

[2] R. D. Authority, "NATIONAL ROAD MASTER PLAN," *Main Report,* 2021.

[3] J. Weerawardana, "GREATER COLOMBO TRAFFIC MANAGEMENT," 2009.

[4] S. Piriyanth, "Sri Lankan traffic congestion," p. 9, 2023.

[5] A Vajeeran, GLDI De Silva, "Identification of Effective Intersection Control Strategies During Peak Hours," 2019.

[6] TWKIM Dias, NHARS Ekanayake, "Evaluating the Efficiency of a Traffic Signal Light over a Traffic Policeman".

[7] E. P. Wannige, "EVALUATING THE IMPACTS OF COORDINATED TRAFFIC SIGNAL SYSTEMS," p. 4, 2018.

[8] I.T. K. JAYASEKARA, R. SANJEEWA, M. T. M. PERERA, "An Automated Traffic Signal System Based on Traffic Queue Length," *IRE Journals,* vol. 2, no. 8, 2019.

[9] D. R. Kodagoda, "Deep Reinforcement Learning to Minimize Traffic Congestion with Emergency Facilitation," 2019.

[10] S M Masfequier Rahman Swapno, Sm Nuruzzaman Nobel, , "Traffic Light Control Using Reinforcement Learning," 2024.

[11] Michael Behrisch, Laura Bieker, Jakob Erdmann, Daniel Krajzewicz, "SUMO – Simulation of Urban MObility An Overview," 2011.

[12] Daniel Krajzewicz, Georg Hertkorn, Peter Wagner, "SUMO (Simulation of Urban MObility) An open-source traffic simulation," 2002.

[13] BEAKCHEOL JANG, MYEONGHWI KIM, GASPARD HARERIMANA, JONG WOOK KIM, "Q-Learning Algorithms: A Comprehensive Classification and Applications," *Digital Object Identifier,* vol. 1109, no. 10, 2019.

[14] C. J. WATKINS, "Technical Note Q-Learning," *Machine Learning,* Vols. 279-292, no. 8, 1992.

[15] M. Sewak, "Deep Q Network (DQN), Dou- ble DQN and Dueling DQN," 2019.

[16] Sven Gronauer1 · Klaus Diepold1, "Multi-agent deep reinforcement learning: a survey," 2021.

[17] DENGYU LIAO1, ZHEN ZHANG 1, TINGTING SONG2, AND MINGYANG LIU1, "An Efficient Centralized Multi-Agent Reinforcement Learner for Cooperative Tasks," 2023.

[18] Teerapun Meepokgit, Sumek Wisayataksin, "Traffic Signal Control with State-Optimizing Deep Reinforcement Learning and Fuzzy Logic," 2024.

[19] Kaiqing Zhang , Student Member, IEEE, Zhuoran Yang , Han Liu , Tong Zhang , and Tamer Ba¸ sar , Life Fellow, "Finite-Sample Analysis for Decentralized Batch Multiagent Reinforcement Learning With Networked Agents," 2023.

file:///Users/chamodkanishka/Library/CloudStorage/OneDrive-kcg.ac.jp/MP2/MP2/Maps/test/edge_types.typ.xml