OPTIMIZING URBAN TRAFFIC FLOW: REINFORCEMENT LEARNING-BASED TRAFFIC LIGHT CONTROL

DEMETRE DZMANASHVILI

Thesis supervisor: ANAIS GARRELL ZULUETA (Department of Automatic Control)

Degree: Master Degree in Artificial Intelligence

Master's thesis

School of Engineering
Universitat Rovira i Virgili (URV)

Faculty of Mathematics
Universitat de Barcelona (UB)

Barcelona School of Informatics (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Abstract

The ever-increasing population as well as the ever-increasing demands placed on transportation are the root causes of the persistent problem of traffic congestion in metropolitan places. In situations with low traffic density, traditional traffic signal systems typically result in inefficiencies and unnecessary delays. These problems are compounded when the signal times are fixed. In order to overcome these obstacles, the study presented here suggests a unique Traffic Control System that is founded on reinforcement learning (RL). We hope that by adding RL, we will be able to construct an intelligent traffic control system that can adapt to the circumstances of the traffic in real time, lessen the impact on the environment, shorten the length of delays, and improve road safety. Our research focuses on a four-way junction, where we examine the incoming traffic density to determine how best to operate the traffic signals and how we may increase the overall efficiency of transportation over time.

The motivation for this research stems from the need to alleviate traffic problems in various regions, including the author's home country, Georgia, and the global demand for intelligent traffic control systems. Rapid urbanization and population growth have placed a substantial burden on urban transportation infrastructure, necessitating innovative and adaptive solutions. By harnessing RL and artificial intelligence, this research aims to contribute to the development of intelligent traffic control systems that can serve as models for cities worldwide.

The primary objectives of this research include the creation of a realistic urban traffic simulation environment, the generation of authentic traffic patterns, the utilization of state-of-the-art RL algorithms, and a comparative analysis of the performance of RL-based controllers against baseline controllers. The research concludes with insights into the potential for intelligent traffic management systems to alleviate congestion, improve efficiency, and reduce environmental impacts.

In summary, this thesis explores the complexities of urban traffic control and the potential of RL-based solutions. It emphasizes the importance of adaptive and context-aware traffic light control systems, offering valuable insights for the optimization of urban traffic flow in real-world settings.

Contents

1. Introduction

1.1 Overview

As a result of an ever-increasing urban population and the rising demand for transportation in cities all over the world, traffic congestion is a chronic global issue that impedes everyday commutes. [13, 26] This is a problem that affects cities all over the world. The delay that is generated by red lights at junctions, which often run on fixed-time schedules regardless of real traffic circumstances [18], is one of the primary factors that contributes to this problem. Although such systems are useful in locations with a high volume of traffic, experience shows that they are frequently ineffective in settings with a low volume of traffic. As a consequence, they cause needless delays and wasteful use of fuel [18].

In light of recent developments in technology, the Adaptive Traffic Signal Control System has been developed. This system makes use of sensors that are installed in roadways in order to synchronize traffic lights, and it does so in order to adapt to real-time traffic circumstances [13]. Despite this, the practicability and cost-effectiveness of this system have been called into question because it requires embedded road infrastructure as well as power sources [13]. In addition, improving traffic signal regulation in order to reduce delays as much as possible while maintaining system reliability continues to be a difficult task [18].

Reinforcement learning (RL) is a framework for artificial intelligence that learns optimum decision policies via continuous adaptation to real-time traffic scenarios. The purpose of this thesis is to solve these issues by developing a Traffic Control System that is based on RL. This system would be used to manage traffic. We want to establish an intelligent traffic control system by moving away from fixed-time schedules and adding RL. This will allow us to efficiently manage traffic flow, decrease environmental impacts like as air pollution and fuel wastage, and improve road safety [18]. This study will concentrate on a four-way junction, examining the density of incoming traffic to determine the best strategy to operate the traffic signals and increase the overall efficiency of transportation over time.

1.2 Motivation

My personal motivation for embarking on this thesis is deeply rooted in the persistent traffic problems that afflict my home country, Georgia. The congestion and inefficiency of traffic lights on some of the busiest streets in Georgia have long been a source of frustration for me and my fellow citizens. The resulting traffic jams not only waste valuable time but also contribute to environmental issues such as increased air pollution and fuel wastage. Moreover, the heightened risk of accidents in congested traffic conditions underscores the urgency of finding effective solutions.

Beyond my personal experiences, the global need for intelligent traffic control systems has never been more evident. Rapid urbanization and population growth have placed an ever-increasing burden on urban transportation infrastructure. As cities around the world grapple with the challenges posed by burgeoning traffic volumes, there is a pressing demand for innovative and adaptive solutions.

In this context, my motivation converges with a broader societal need for intelligent traffic light systems. These systems have the potential to revolutionize urban transportation by dynamically managing traffic flows, reducing congestion, and mitigating environmental concerns. By harnessing the power of reinforcement learning and artificial intelligence, I aim to contribute to the development of intelligent traffic control systems that can serve as a model for cities worldwide.

Through this research endeavor, I aspire to make a meaningful impact by fostering more efficient and sustainable urban transportation systems. By optimizing traffic light control, I seek not only to alleviate the traffic woes in my homeland but also to offer a scalable solution that addresses the global imperative for intelligent traffic management.

1.3 Objectives

The key goals of this thesis are doing an in-depth examination of enhancing urban traffic flow using a strategy that incorporates a variety of different perspectives. These goals are meant to address the difficulties of traffic management, increase the level of realism in simulations, and evaluate the effectiveness of both cutting-edge algorithms and baseline controllers. The following is a list of the primary goals:

Creation of Vake Map in SUMO

The first objective is to develop a complex and representative urban traffic simulation environment within the Simulation of Urban MObility (SUMO) framework. This entails the creation of a detailed Vake map, capturing the intricacies of traffic infrastructure, including road networks, intersections, and traffic lanes. The map should accurately reflect the real-world urban environment under investigation.

Generation of Realistic Traffic Patterns

It is absolutely necessary to use actual traffic patterns in order to make the simulations as realistic as possible. In order to accomplish this goal, you will need to gather data on the traffic situation at the specified site in real time and painstakingly document the timing and behavior of the traffic lights. After that, the data that was acquired will be included into the simulation environment in order to imitate the actual circumstances of the traffic.

Utilization of State-of-the-Art Algorithms

The core of this research lies in the exploration, implementation and adaptation of state-of-the-art traffic signal control algorithms. The following algorithms will be employed:

- **IDQN**: Implementing this deep reinforcement learning algorithm for traffic signal control, which has shown promise in optimizing signal timings.
- **IPPO**: Utilizing IPPO as another reinforcement learning algorithm to investigate its effectiveness in traffic management.
- **MPLIGHT**: Exploring MPLIGHT, a multi-phase control algorithm designed to adapt traffic signals dynamically.
- **FMA2C**: Investigating the potential of FMA2C for cooperative multi-agent traffic signal control.

Comparison with Baseline Controllers

To evaluate the performance of the selected state-of-the-art algorithms, this objective involves implementing and assessing the following baseline controllers:

- **Fixed Time Control**: A traditional control strategy with fixed signal timings that do not adapt to real-time traffic conditions.
- **Max-Pressure Control**: Implementing this controller, which focuses on minimizing congestion by prioritizing the most congested lanes at intersections.
- **Greedy Control**: Assessing the performance of a basic greedy controller that makes decisions based on immediate traffic conditions.

Comparative Analysis and Conclusions

After the simulations and experiments have been run to completion, the data obtained from the various traffic signal control algorithms and baseline controllers will be meticulously examined and contrasted with one another. The goal is to arrive at significant findings on the effectiveness of each strategy in maximizing the flow of urban traffic. The purpose of this study is to shed light on the possibilities for intelligent traffic management systems to decrease environmental consequences, enhance efficiency, and alleviate congestion.

Through the accomplishment of these goals, the purpose of this thesis is to make a significant intellectual contribution to the subject of urban traffic optimization and to give actionable recommendations for improving traffic signal management systems in actual urban environments.

2. Background

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a paradigm in which an agent learns to make decisions by interacting with its environment. In the RL framework, the environment is often modeled as a Markov decision process (MDP), characterized by key components:

- S - the state space,
- A - the action space,
- $P(s_t, a, s_{t+1})$ - the transition function, mapping from state s_t and action a to the next state s_{t+1} with probabilities in the range [0, 1],
- $R(s, a)$ - the reward function, which assigns a real-valued reward to each state-action pair,
- γ - the discount factor, controlling the trade-off between immediate and future rewards.

The RL agent operates based on a policy π, which maps states to actions, i.e., $\pi : S \to A$. When the agent selects an action a_t in the current state s_t, it impacts the environment, leading to a new state s_{t+1} and an immediate reward r_t.

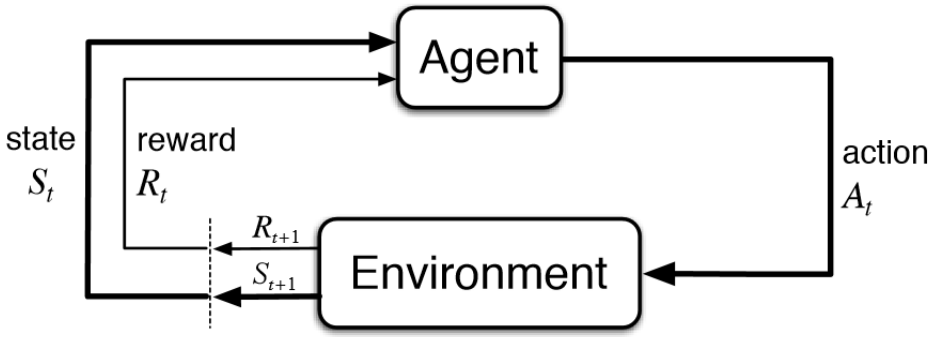


Figure 1: Reinforcement Learning Framework

The primary objective of the RL agent is to maximize the expected sum of discounted rewards, denoted as $J^\pi = \sum_{t=0}^{\infty} \gamma^t r_t$. The optimal policy, denoted as π^*, is the one that maximizes this objective.

There are various approaches for training a policy using RL:

- **Value-Based Approach:** This approach focuses on estimating the expected future utility from states (state value) or from action-state pairs (action value or q-value). The control policy is then directed towards actions or states that maximize the expected utility (J^π). A prominent example is the model-free deep Q-learning algorithm [17].

- **Policy-Gradient Approach:** In this approach, a policy is defined through a parameterized differential equation, and the parameters are updated incrementally following the policy gradient. These updates aim to achieve favorable outcomes as measured by the reward function. Estimations of state or action values are often used to define these favorable outcomes. This approach is commonly referred to as an actor-critic approach.

- **Actor-Critic Approach:** Actor-critic methods combine elements of both value-based and policy-gradient approaches. An actor (policy) learns to make decisions, while a critic (value function) evaluates these decisions. A state-of-the-art example of an actor-critic algorithm is the proximal policy optimization (PPO) algorithm [21].

These RL approaches provide a framework for training intelligent agents to make decisions in complex and dynamic environments, making them highly relevant to optimizing traffic signal control in urban settings.

2.2 Traffic Signal Control as an MDP

In the realm of traffic engineering, a signalized intersection represents a complex network of incoming and outgoing roads, each comprising one or more lanes. To efficiently manage traffic flow at such intersections, a set of phases, denoted as Φ, is defined. Each phase, $\varphi \in \Phi$, corresponds to a specific traffic movement through the intersection, as illustrated in Figure 2. It's crucial to note that two phases are considered conflicting if they cannot be simultaneously enabled due to intersecting traffic movements.

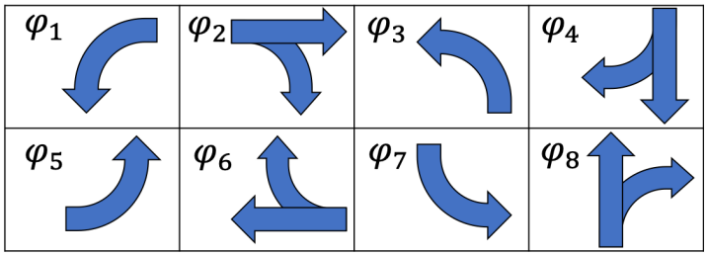


Figure 2: Example of Phases at a Signalized Intersection[4]

A signal controller's job is to choose, at each discrete time step, whether combinations of phases that do not interfere with one another should be enabled. The purpose is to maximize a long-term objective function, which can take on a variety of forms based on the particular requirements imposed by unique goals. When discussing controllers that are based on Reinforcement Learning (RL), it is usual practice to represent the signalized intersection environment as a Markov Decision Process (MDP). This model includes the following components:

- **State Space (S):** The state space encompasses the state of incoming traffic and the currently enabled phases. The definition of the state varies among studies, reflecting differing sensing capabilities. Some works assume state-of-the-art traffic sensing technologies, providing high-resolution data on incoming traffic, including information such as the number of approaching vehicles, accumulated waiting time, the number of stopped vehicles, and the average speed of approaching vehicles [5]. Others adopt less informative sensing capabilities, such as observing only the stopped queue length per lane [16] or solely the waiting time of the first vehicle in the queue [22].

- **Action Space (A):** In each time-step, the controller selects a set of non-conflicting phases to be assigned the right-of-passage (green light). If the chosen phases are different from the currently enabled ones, a mandatory yellow phase is enforced ny the system for a predefined duration. It's important to note that assigning yellow phases is not the part of the action space, it is a constraint imposed by the environment.

- **Transition Function (P):** The transition function describes the progression of traffic following the signal assignment. This progression can be defined within a simulated environment, as commonly done in research [16], or based on real-world traffic progression in practical implementations.

- **Reward Function (R):** The reward function serves as a critical component in RL-based signal control. Different reward functions have been proposed in the literature. Commonly used reward functions include (minus) queue length summed over all incoming lanes [29], (minus) total delays imposed by the intersection [22], (minus) waiting time at the intersection [16], and (minus) traffic pressure [5]. These reward functions reflect various aspects of traffic performance and congestion alleviation.

The modeling of traffic signal control as an MDP provides a foundation for applying RL techniques to optimize signal operation, ultimately contributing to more efficient and adaptive traffic management strategies.

2.3 Evaluation environments for RL-based signal controllers

According to [4], previous research in the field of traffic signal control has often relied on custom-made scenarios tailored for evaluating specific Reinforcement Learning (RL) algorithms. For instance, Jinming and Feng (2020) utilized the well-established Simulation of Urban Mobility (SUMO) environment for their experiments. SUMO enjoys widespread acceptance within the transportation community and serves as a reasonable testbed choice for such studies. However, it's worth noting that Jinming and Feng's reported scenario, based on the real-world city of Monaco, was a modified version. This modified scenario included 18 synthetic traffic signals beyond the official "Monaco SUMO Traffic (MoST)" scenario and incorporated non-validated inflated traffic demands [7].

Another notable simulation testbed, CityFlow, was presented by Zhang et al.[30]. However, CityFlow has two primary limitations. Firstly, unlike SUMO, CityFlow lacks rigorous calibration and evaluation within the general transportation community. Although it claims to produce equivalent output as SUMO, this claim is primarily based on results from simplified grid network scenarios. Secondly, while CityFlow offers the Manhattan, New York network as a common benchmark scenario, the support for this scenario's representation of real-world city layouts and demands is limited.

Additionally, some relevant publications have conducted evaluations using the Autonomous Intersection Management (AIM) simulator. The primary drawback of the AIM simulator lies in its lack of traffic scenarios based on real-world cities. AIM typically generates simple grid networks with symmetric intersections. While one might draw parallels between such grid networks and the road layout in Manhattan, New York, a more in-depth analysis of traffic trends is needed to substantiate such claims and their relevance to the real world [20][9][16].

3. Related Work

3.1 Reinforced Signal Control (RESCO)

In this section, we review related work in the field of traffic signal control, with a focus on the Reinforced Signal Control (RESCO) toolkit, which serves as a baseline for my research.

The RESCO toolkit is a standard Reinforcement Learning (RL) traffic signal control testbed designed to achieve several key objectives:

1. Provide benchmark single and multi-agent signal control tasks based on well-established traffic scenarios.
2. Offer an OpenAI GYM interface within the testbed environment to facilitate the deployment of state-of-the-art RL algorithms.
3. Deliver a standardized implementation of state-of-the-art RL-based signal control algorithms.

RESCO is open-source and freely available under the GNU General Public License 3. It is built on top of SUMO-RL [2] and can be accessed on GitHub at `github.com/Pi-Star-Lab/RESCO`. The embedded traffic scenarios within RESCO have their own licensing, with Cologne-based scenarios under Creative Commons BY-NC-SA and Ingolstadt-based scenarios under the GNU General Public License 3.

3.1.1 State and Action Space

RESCO is able to accept a broad spectrum of sensing assumptions, including sophisticated sensing capabilities [7]. Users are able to make feature selections for the state that are based on certain sensing assumptions. At the level of the state, junction, and lane, the features include information about the length of the line of stopped cars, the number of vehicles that are approaching, the total amount of time that stopped vehicles have been waiting, and more [7]. In addition, users have the ability to define the effective sensing distance when the system is initialized.

Following the technique outlined in Section 2.2 of the RESCO handbook [7], the action space in RESCO includes sets of phase combination options that do not cause a conflict with one another. Actions are selected for the subsequent ten seconds of the simulation by default, with the first three seconds being held in reserve for yellow signals, if they are required.

3.1.2 Reward Metrics

RESCO offers flexibility in terms of reward metrics. Users can designate any of the reward metrics defined in Section 2.2 of the RESCO documentation[7] or create custom weighted combinations of these metrics. When initializing a control task, users can pass a weight vector that assigns weights to different metrics in the reward function. These weights correspond to various aspects, such as system travel time, signal-induced delays, total waiting time at intersections, average queue length, and traffic pressure.

3.1.3 Benchmark Control Tasks

The signal control benchmark tasks in RESCO are based on two well-established SUMO scenarios: "TAPAS Cologne" and "InTAS" [20, 14]. These scenarios represent traffic within real-world cities, namely, Cologne and Ingolstadt in Germany. They include road network layouts and calibrated demands, making them suitable for comprehensive evaluation. RESCO defines three benchmark control tasks for each traffic scenario:

1. Controlling a single main intersection.
2. Coordinated control of multiple intersections along an arterial corridor.
3. Coordinated control of multiple intersections within a congested area (downtown).

3.1.4 Benchmark Algorithms

RESCO provides three baseline controllers and several RL-based controllers for comparative evaluation:

1. **Baseline Controllers**:
   (a) Fixed-time (Pre-timed) control, where phase combinations are enabled for fixed durations following predefined cycles, that was recorded physically from the real-world traffic signal controller.
   (b) Max-pressure control, which selects the phase combination with the maximum joint pressure. [5]
   (c) Greedy control, which chooses the phase combination with the maximum joint queue length and approaching vehicle count.[16]
2. **RL Controllers**:
   (a) IDQN (Independent DQN agents), employing convolutional layers for lane aggregation[3].
   (b) IPPO, which utilizes a deep neural network similar to IDQN[3].
   (c) MPLight, based on the FRAP open-source implementation, ChainerRL DQN[10], and pressure sensing[32].
   (d) Extended MPLight (MPLight*), an enhanced version of MPLight with additional sensing information.
   (e) FMA2C, built on top of the MA2C open-source implementation[6].

In each of the RL-based controllers, specific learning algorithms and hyperparameters are applied, allowing for a comprehensive evaluation of their performance [3, 5, 6, 16, 32].

In the case of IDQN, IPPO, and MPLight, the implementation of the learning algorithm is invoked directly from the ChainerRL [10] and the Preferred RL [10] libraries that is successor of ChainerRL, and customized to align with my specific map and requirements.

3.2 Reinforcement-Learning-Light (RLight)

Adaptive Traffic Signal Control (ATSC) is a critical aspect of modern transportation systems, with the primary goal of enhancing traffic flow, reducing travel times, and mitigating CO2 emissions [31]. Traditional methods of traffic signal control often rely on manually crafted rules that cater to specific traffic scenarios. However, recent advancements in traffic data collection and optimization techniques suggest new possibilities [28].

In recent years, the integration of deep learning and reinforcement learning (RL) has shown remarkable success in solving complex tasks, often achieving super-human performance in various domains [11, 17, 23]. This success raises the question of whether deep RL can bring similar benefits to ATSC.

Yet, ATSC poses a unique challenge from an RL perspective. Unlike many RL applications where the environment naturally maps to a Markov Decision Process (MDP), ATSC lacks a clear source of

raw data and rewards, and it doesn't have a fixed set of actions or action rates. The quality of the MDP representation plays a crucial role in the success of ATSC [33].

Previous models that aim to provide agents with comprehensive information often result in unnecessary complexity that hampers performance [33]. In contrast, Light-Intellight (LIT) [33] proposes a minimal set of features based on a uniform traffic distribution, demonstrating effective performance. However, urban traffic often exhibits non-uniform, clustered patterns, as depicted in Figure 3. Surprisingly, prior work has not specifically tailored their state representations to address this clustered traffic data.

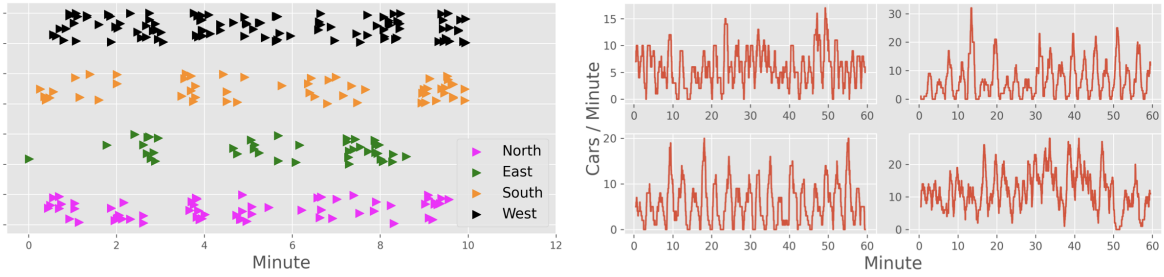


Figure 3: The left panel shows data from the first ten minutes of inflow on intersection Hangzhou 2, with every triangle indicating a vehicle. The right panel shows the rate of vehicles per minute. [12]

Reinforcement-Learning-Light (RLight)[12] is a simplistic yet effective reinforcement learning framework that was created to exploit inductive biases that are peculiar to ATSC. In order to solve these issues, the authors present this framework. They improve the state representation in order to accept clustered traffic data by capitalizing on the capabilities that LIT has. In their method, they differentiate between cars that are approaching and those that are waiting, and they describe the condition of the system by factoring in the mean speed and distance of the vehicles that are coming. This approach makes use of the inherent structure that traffic clusters have in order to maintain a representation of the state that is succinct and actionable. Their method bridges the gap between RL and real-world applications of ATSC by making use of structured data sources like ground sensors and GPS data.

It is important to note that the modeling of yellow light in ATSC has been mainly neglected in previous studies. They study the prospect of introducing inductive biases that can skip redundant time steps during yellow lights, thereby simplifying the reward signal and expediting the learning process by about one-third. While the issue has usually been framed as a normal MDP, this research investigates the possibility of incorporating inductive biases.

In addition, there is no agreement among the researchers who have done work in the past about whether or not cyclic or acyclic action spaces are more secure and effective. They study this distinction as well, offering insight on the implications of letting the agent to freely pick the ideal phase at each time step in the process.

They evaluate their approach using real data from five intersections in Hangzhou, employing the CityFlow simulator [30] unlike the simulator that I used in this work. To ensure the generalization of their method, they split the dataset into training, validation, and test sets. In their experiments, they compare RLight to a rule-based method known as Self-Organizing Traffic Lights (SOTL) [8] in both multi-phase and two-phase settings. According to them they achieve outstanding results where RLight consistently outperforms state-of-the-art methods across all five intersections.

In summary, their contributions in this research are as follows:

- Introduction of RLight, a straightforward RL framework that exploits inductive biases relevant to ATSC. Their cluster-aware state representation outperforms existing methods on the Hangzhou dataset.
- Proposal of reformulation of the MDP that eliminates redundant yellow light time steps, significantly expediting the learning process.
- Investigation of the performance difference between acyclic and cyclic phase transitions, offering valuable insights into the design of action spaces.

- Extension of the rule-based SOTL algorithm to work in an acyclic manner for multi-phase intersections.

According to them their single-agent approach demonstrates the capability to effectively control varying traffic densities, making it adaptable to unexpected real-world scenarios, including events, accidents, or unforeseen situations like a pandemic.

3.2.1 RLight Agent Design

Their RLight agent is based on the standard RL framework [24]. In the following sections, we will briefly describe how they formulate ATSC as an MDP and delve into the state and action representation, as well as the reward function. They consider scenarios where the agent controls a single intersection with J incoming lanes and I traffic light phases.

3.2.2 Markov Decision Process

In ATSC problems, the modeling of the action rate has received limited attention in previous research. They consider two options based on the timestep rate of one simulation second per transition and a fixed yellow light period.

MDP (Markov Decision Process): In this approach, the agent selects an action at each simulation second, but actions during yellow light are ignored by the environment. This means the agent must learn that actions during yellow states have no impact on the final reward. Consequently, training effort is wasted on learning irrelevant state-action pairs, potentially introducing noise into the reward signal.

SMDP (Semi-Markov Decision Process): In this scheme, the agent only chooses actions when its decisions affect the environment, remaining inactive during yellow light. This can be seen as a Semi-Markov Decision Process (SMDP) [25]. The yellow period has a fixed length of ψ timesteps. When the agent switches to another phase at timestep τ, it receives cumulative discounted rewards during the yellow period, enhancing learning efficiency.

State Representation

The state representation is a vital aspect of ATSC. At each time step, the agent receives a quantitative representation of the environment, which they aim to make easily digestible while containing the necessary information for decision-making.

Starting with the number of vehicles and the current phase, as used in LIT [33], they introduce:

$$\boldsymbol{s}_t = [\boldsymbol{w}_t^\intercal + \boldsymbol{a}_t^\intercal, \boldsymbol{p}_t^\intercal],$$

where $\boldsymbol{w} + \boldsymbol{a} \in \mathbb{R}^J$ represents the total number of vehicles (waiting, w, plus approaching, a) on each lane, and $\boldsymbol{p}$ is the phase represented as a one-hot vector of size I, where I and J are the number of phases and lanes, respectively.

However, this simple approach may lead to indistinguishable states, hindering learning. To address this, they explicitly separate waiting and approaching vehicles, catering to the fragmented distribution of urban traffic. Additionally, they incorporate the average speed and distance of approaching traffic to improve traffic anticipation, enabling earlier phase switches if a cluster moves faster or closer.

This enhanced state representation becomes:

$$\boldsymbol{s}_t = [\boldsymbol{w}_t^\intercal, \boldsymbol{a}_t^\intercal, \boldsymbol{d}_t^\intercal, \boldsymbol{s}_t^\intercal, \boldsymbol{p}_t^\intercal],$$

where $\boldsymbol{w}$ represents the number of waiting vehicles, $\boldsymbol{a}$ the number of approaching vehicles, $\boldsymbol{d}$ is the average distance of approaching vehicles, $\boldsymbol{s}$ is the average speed of approaching vehicles, and $\boldsymbol{p}$ is the phase represented as a one-hot vector, with all values normalized. This method assumes that vehicles behave like steady convoys, leading to a state-space dimension of $4 \times J + I$.

Action Space

At each timestep t, the agent selects an action a_t from the available set of actions $\mathcal{A} = \{1, \ldots, K\}$. They aim to provide the agent with maximum freedom to choose the most suitable action. They explore two options:

Cyclic: In this configuration, we use a predetermined phase sequence where the agent can either keep the current phase or switch to the next phase. The neural network outputs a value to indicate whether to switch or stay.

Acyclic: In the acyclic setup, the agent can freely choose the next phase, offering more flexible control. Here, the network outputs as many values as there are phases.

Reward Function

In ATSC, the agent receives a numerical reward r_t at each timestep, which is defined by the reward function. Their aim is to formulate a reward function that minimizes average travel time, assuming it as the primary objective.

Average travel time is challenging to compute in real-time, and using it directly as the reward signal would result in sparse and delayed rewards. Therefore, they employ a shaping reward based on the total queue length at the intersection [33]:

$$\mathcal{R}(s) = -\sum_j w(s)_j t_j,$$

where w_j is the queue length on lane j and j is the number of lanes. This shaping reward is proportionally related to the average travel time when speed changes are neglected.

3.2.3 Self-Organizing Traffic Lights 2.0

In this section, we will discuss their adaptation for the rule-based Self-Organizing Traffic Lights (SOTL) algorithm to accommodate multi-phase settings. The original SOTL was primarily designed for two-phase settings, where the agent would switch to the next phase. However, in multi-phase intersections, the problem becomes more complex as multiple phases can exceed the threshold simultaneously. To address this, they introduce modifications to the algorithm.

In the original implementation, a key issue arises when transitioning to a multi-phase intersection. The resetting of a parameter, denoted as κ, to zero no longer sufficiently conveys that vehicles have passed through a green phase. This is particularly problematic when multiple phases share lanes. If one phase turns green, the vehicles on the shared lane will pass through it, affecting κ for that phase. However, κ for other phases sharing the lane remains unaffected. As a result, the phase with the most cumulative waiting time is chosen, even if no vehicles are waiting for it. To overcome this, they introduce a new parameter, ρ, which represents a set of counters to track vehicle integrals per lane over time. When a lane gets a green light, all corresponding counters are reset, ensuring that passed vehicles are removed from all relevant integrals.

In this extended SOTL algorithm, the phase with the maximum cumulative waiting time, denoted as κ, is chosen. The algorithm's parameters include the current duration of the phase ϕ_{green}, the minimal phase duration $\phi_{\min}$, the number of vehicles within a specified distance from green lights $v^{\text{green}}_{\text{vehicles}}$, a tunable parameter μ that determines the number of vehicles needed to split a cluster, and the set of counters ρ corresponding to each phase ϕ_i.

This improved method guarantees that the phase with the longest waiting time is picked while also solving the issue of passing cars in multi-phase junctions. It does this by ensuring that the phase with the most waiting time is selected. It is important to note that the updated algorithm functions efficiently in two-phase, four-approach situations, just like the original SOTL technique.

3.3 Diagnosing Reinforcement Learning for Traffic Signal Control

In this section, we discuss relevant study that was done for diagnosing some of the Reinforcement Learning algorithms for Traffic Signal Control. [33]. We will provide some brief overview of their

Algorithm 1 SOTL Generalized to Multi-Phase Settings[12]

```
Initialize κ and ρ to 0
for t from 1 to T do
    for each lane j do
        ρ += ν_j
    end for
    for each phase i do
        Compute κ_i according to the integrals ρ and the phase duration φ_i
    end for
    if φ_green > φ_min then
        Determine the phase with the highest κ
        Choose the corresponding action
    end if
end for
```

work. Most of the formulas will be directly cited from their work to have understanding of the their interception of the Reinforcement Learning algorithms for Traffic Signal Control.

3.3.1 Problem Formulation

The specification of the RL environment is the basis of the issue that arises while attempting to optimize traffic signal control by utilizing RL. Both the representation of the state and the mechanism by which decisions are made are extremely important in this situation. The RL agent monitors the environment, which is encapsulated by the numerical state representation s_t, and then decides whether or not to advance the signal to the next phase by keeping the signal that is now active, which is symbolized by the symbol a_t. This idea of a pre-defined signal phase order is consistent with well-established methods in transportation engineering[25, 27], which helps to ensure that driver expectations and safety concerns are taken into consideration.

Formal Problem Statement

The core problem addressed in these studies can be formally defined as follows:

Problem 1. Given the state observations set S, action set A, and the reward function $R(s, a)$, the problem is to learn a policy $\pi(a|s)$ that maximizes the expected discounted return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{b=0}^{\infty} \gamma^b R_{t+b+1}$$

Key components of this problem include state representations, action decisions, and reward structures.

State, Action, and Reward

The state representation typically includes information about the number of vehicles on each lane ($v_{j,t}$) and the current signal phase (p_t). The action is binary, where the signal can either change ($a_t = 1$) or remain the same ($a_t = 0$). The reward is defined as the summation of queue length over all lanes:

$$R_t = -\sum_{j=1}^{M} q_{t,j}$$

Here is a summary of the key notation used in these RL-based traffic signal control studies (Table 1).

Notation	Meaning
s	State
a	Action
R	Reward
q_j	Queue length on lane j
v_j	Number of vehicles on lane j
p	Signal phase
K	Number of signal phases
M	Number of lanes
N	Number of vehicles in the system

3.3.2 Objective

The primary objectives of the research in this area can be summarized as follows:

- Find an RL algorithm to effectively address the traffic signal control problem.
- Establish connections between RL algorithms and classic transportation theory, demonstrating the optimality of the RL approach.
- Analyze the traits that make RL outperform other traditional methods in traffic signal control.

This review sets the stage for our proposed RL approach and its contributions to solving the traffic signal control problem.

Hello there

3.3.3 Methodology

Several studies have delved into traffic signal control problems using RL. A key development in this field is the Light-IntelligT (LIT) method, which is designed to address the traffic signal control problem.

Deep Q-Network

LIT leverages the Deep Q-Network (DQN) to seek actions that maximize long-term rewards. The Bellman Equation [25] serves as a fundamental principle in this approach:

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma \max Q(s_{t+1}, a_{t+1})$$

The use of DQN, as illustrated in Figure 4, demonstrates the effective use of features such as vehicle numbers and phase indicators.

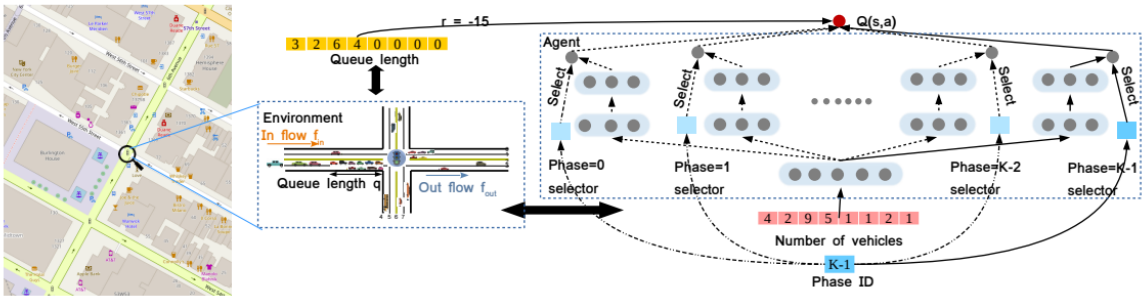


Figure 4: Overview of their traffic signal control system.[33]

Classic Transportation Theory

Before delving into RL-based traffic signal control algorithms, it's important to consider traditional transportation theory. The optimization of traffic signal control from a transportation research perspective focuses on minimizing total travel time under specific traffic conditions.

The mathematical expression for this optimization problem involves constraints related to green phase allocation and traffic flow:

$$\begin{array}{ll} \text{minimize} & \sum_{j=1}^{M} T_j \\ \text{subject to} & g^{\min} \leq g_k \leq g^{\max} \\ & \frac{g_j}{C} \geq \frac{f_{in,j}}{u_{sat,j}}, \quad j = 1, 2, 3, \ldots, M \end{array}$$

The constraints ensure appropriate green phase allocation and traffic flow considerations. This traditional transportation approach lays the foundation for understanding the traffic signal control problem from a different perspective.

Connecting RL with Transportation Theory

This section aims to build connections between the RL algorithm and transportation methods. It highlights the equivalence between using queue length as a reward function in RL and optimizing travel time in transportation methods. Additionally, the use of the number of vehicles on each lane (v_j) and the current phase (p) as the only state features is discussed, emphasizing their importance in understanding system dynamics.

Analysis of Traits of RL Approach

The analysis of RL's traits that enable it to outperform traditional methods is another critical aspect of these studies. These traits include:

- Online learning: RL algorithms adapt in real-time based on feedback from the environment, learning from mistakes.
- Sampling guidance: RL employs experience to make guided decisions, improving convergence and reward.
- Forecast: RL predicts future rewards through the Bellman equation, which enables actions for long-term rewards.

These traits distinguish RL as a powerful approach in traffic signal control.

4. State of the Art

4.1 IDQN

Reinforcement Learning (RL) is a prominent area of machine learning where agents learn to make sequential decisions by interacting with an environment. DQN, short for Deep Q-Network, is a fundamental algorithm in RL that leverages deep neural networks to approximate optimal action-value functions.

4.1.1 Deep Q-Network (DQN)

DQN, proposed by Mnih et al. [17], is designed to address the challenges of learning Q-values in high-dimensional state spaces. It combines Q-learning, a well-established RL algorithm, with deep neural networks.

The Q-value, denoted as $Q(s, a)$, represents the expected cumulative reward when taking action a in state s. DQN approximates this Q-value using a deep neural network with parameters θ. The Q-network is trained to minimize the temporal difference (TD) error:

$$\delta = Q(s, a; \theta) - (r + \gamma \max_{a'} Q(s', a'; \theta^-))$$

Where:

δ - TD error

$Q(s, a; \theta)$ - Q-value predicted by the network

r - Immediate reward

γ - Discount factor

$Q(s', a'; \theta^-)$ - Target Q-value predicted by a target network with parameters θ^-

DQN employs experience replay and a target network to stabilize training. Experience replay stores past experiences in a replay buffer and samples mini-batches for training, breaking the temporal correlation in the data. The target network provides stable target Q-values for the TD error.

4.1.2 Independent Deep Q-Networks (IDQN)

IDQN is an extension of DQN tailored for multi-agent RL scenarios, where multiple agents operate independently to optimize their actions. Each agent in IDQN maintains its own Q-network and replay buffer.

The Q-value update rule in IDQN remains similar to DQN, but it is extended to accommodate multiple agents:

$$\delta = Q_i(s, a_i; \theta_i) - (r + \gamma \max_{a'} Q_i(s', a'; \theta^-))$$

Where:

δ - TD error for agent i

$Q_i(s, a_i; \theta_i)$ - Q-value predicted by agent $i's$ network

r - Immediate reward

γ - Discount factor

$Q_i(s', a'; \theta^-)$ - Target Q-value predicted by agent $i's$ target network

IDQN facilitates decentralized decision-making among multiple agents, making it suitable for scenarios involving cooperation or competition among agents.

To explore IDQN in more detail, the following paper[3] provide comprehensive insights into its theory and applications

4.2 IPPO

Proximal Policy Optimization (PPO) is a state-of-the-art reinforcement learning algorithm designed for optimizing parameterized policies in complex environments. IPPO, short for Independent Proximal Policy Optimization, is an extension of PPO tailored for multi-agent reinforcement learning scenarios, where multiple agents learn independently.

4.2.1 Proximal Policy Optimization (PPO)

Introduced by Schulman et al. [21], PPO addresses several challenges in policy optimization. It aims to maximize the expected cumulative reward while ensuring that policy updates are not too large, preventing catastrophic policy changes. PPO achieves this through the following objectives:

Objective Function

PPO optimizes a surrogate objective function that balances the trade-off between policy improvement and policy constraint. The objective function is given as:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right]$$

Where:

$\mathcal{L}(\theta)$ - Surrogate objective function

θ - Policy parameters

$r_t(\theta) = \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ - Importance ratio

$\hat{A}_t$ - Advantage estimate

ϵ - Clip parameter

PPO optimizes this objective function using stochastic gradient ascent.

Trust Region

PPO introduces a trust region constraint by clipping the surrogate objective. The clip function ensures that policy updates do not deviate significantly from the previous policy:

$$\text{clip}(x, a, b) = \begin{cases} x, & \text{if } x \in [a, b] \\ a, & \text{if } x < a \\ b, & \text{if } x > b \end{cases}$$

PPO efficiently balances policy updates to ensure stability and improved performance.

4.2.2 Independent Proximal Policy Optimization (IPPO)

IPPO extends the PPO algorithm for multi-agent RL scenarios, where multiple agents learn independently. Each agent in IPPO maintains its own policy and operates in the environment. IPPO's objective function for agent i remains similar to PPO:

$$\mathcal{L}_i(\theta_i) = \mathbb{E}\left[\min\left(r_t(\theta_i)\hat{A}_t^i, \text{clip}\left(r_t(\theta_i), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^i\right)\right]$$

Where:

$\mathcal{L}_i(\theta_i)$ - Surrogate objective function for agent i

θ_i - Policy parameters for agent i

$r_t(\theta_i) = \dfrac{\pi_{\theta_i}(a_t^i|s_t)}{\pi_{\theta_{i_{\text{old}}}}(a_t^i|s_t)}$ - Importance ratio for agent i

$\hat{A}_t^i$ - Advantage estimate for agent i

IPPO facilitates decentralized learning among multiple agents, making it suitable for scenarios involving independent agents with their policies.

To explore IPPO in more detail, the following paper[3] provide comprehensive insights into its theory and applications

4.3 MPLight

The MPLight[5] traffic light control system is one that makes effective use of the idea of pressure in order to organize the flow of traffic through many intersections. In order for it to function properly, it takes into account the pressure, which can be defined as the disparity between the lengths of the lines waiting to enter a junction and the line waiting to enter an intersection farther downstream. MPLight is intended to improve the flow of traffic in metropolitan areas while simultaneously lowering levels of congestion.

Within MPLight, pressure is utilized as an essential parameter for the purpose of coordinating traffic signals. It is determined by subtracting the length of the line formed by cars waiting to enter a junction from the length of the line formed by vehicles waiting to enter the receiving lane of the intersection farther downstream. MPLight seeks to achieve a balance in the traffic load at numerous junctions by taking pressure into consideration.

MPLight is a method for controlling traffic lights that was developed by Chen et al., and it makes use of principles from reinforcement learning. In order to arrive at judgments about the traffic signals, they made use of Deep Q-Networks (DQN) as the underlying architecture. One DQN agent serves as the point of contact for all of the intersections in this configuration.

In MPLight, pressure is employed not only as a coordination metric but also as the state and the reward for the DQN agent. This is because pressure is a measure of how well the agents are working together. At every particular time step, information about the pressure values at all relevant intersections is included in the state of the agent that is being described. The DQN agent's capacity to learn is facilitated by the reward signal, which is derived from changes in pressure and serves as a guide for that process.

When compared to the approaches that are currently being used, Chen et al.[5] showed substantial improvements in both the flow of traffic and the journey times when MPLight was used. To be more specific, MPLight was able to produce up to a 19.2 percent improvement in travel times when compared to the next best approach, which was PressLight.

4.4 FMA2C

FMA2C[6] is an advanced approach to traffic signal control that utilizes a hierarchical framework to optimize traffic flow in urban environments. It builds upon the prior work of MA2C (Multi-Agent Advantage Actor-Critic) by introducing managing agents to coordinate and oversee workers responsible for signal control at intersections.

4.4.1 Basic Concepts

Workers (Intersection-Level Agents)

In FMA2C, the core agents responsible for signal control at intersections are called workers. Each worker operates independently as an advantage actor-critic agent. The workers are tasked with making real-time decisions regarding traffic signal timings at their respective intersections.

Managing Agents (Region-Level Agents)

In FMA2C, managing agents are introduced, and in comparison to employees, they function at a higher level of the organizational structure. Within the traffic network, the responsibility for a particular region or area falls on the shoulders of each managing agent. These management agents are in charge of many personnel and are tasked with increasing the efficiency of traffic flow within the regions that they are responsible for.

4.4.2 Hierarchical Reinforcement Learning

FMA2C leverages hierarchical reinforcement learning to improve traffic signal coordination. The hierarchy involves two levels: managing agents at the top level and workers at the lower level.

Managing Agent Training

Training is provided to managing agents so that they can maximize the flow of traffic within the regions to which they are allocated. They are given high-level aims and objectives relating to traffic, such as reducing congestion as much as possible or increasing the amount of traffic that can pass through an area. The governing agents will base their judgments at the regional level on these aims.

The training of managing agents can be formulated as a reinforcement learning problem, where the managing agent learns a policy π_m to maximize a region-specific objective function:

$$J_m(\pi_m) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t^m\right]$$

Where:

$J_m(\pi_m)$ - Expected cumulative reward for managing agent m
π_m - Policy of managing agent m
γ - Discount factor
R_t^m - Region-specific reward at time step t

Worker Training

Workers, on the other hand, are trained to incorporate the high-level goals set by their respective managing agents into their local decision-making process. This hierarchical training ensures that workers align their actions with the broader objectives of traffic flow optimization.

The training of workers also involves reinforcement learning, where each worker learns a policy π_w to maximize its intersection-specific objective function:

$$J_w(\pi_w) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t^w\right]$$

Where:

$J_w(\pi_w)$ - Expected cumulative reward for worker w
π_w - Policy of worker w
γ - Discount factor
R_t^w - Intersection-specific reward at time step t

4.4.3 Coordination Mechanisms

In order to maintain efficient traffic signal control, FMA2C makes use of a variety of procedures that facilitate collaboration between managing agents and personnel. The transmission of high-level goals, the distribution of rewards, and coordination through a central mechanism are some examples of the methods that may be used.

4.4.4 Performance Improvement

Through the implementation of a hierarchical structure that enables coordinated decision-making at both the regional and junction levels, FMA2C intends to accomplish the goals of enhancing traffic flow while simultaneously decreasing congestion. FMA2C's purpose is to optimize the timing of traffic signals in an effective manner by ensuring that the activities of workers are aligned with the goals of management agents.

5. Specification of the solution

5.1 Simulation Engines

Simulation is a cornerstone in understanding and optimizing urban traffic flow, playing a central role in my research on optimizing traffic light control through reinforcement learning. In this section, we emphasize the significance of simulating urban mobility, highlighting key components and methodologies.

In the context of my study, conducting real-world experiments to investigate traffic dynamics and assess traffic light control strategies can be impractical and costly. Simulation offers a safe, efficient, and cost-effective alternative.

Simulation allows us to create virtual replicas of urban environments, accurately modeling traffic conditions, vehicle behaviors, and interactions between various elements. This enables researchers to observe and analyze traffic patterns, congestion, and the outcomes of different control strategies without resorting to physical experiments.

Urban mobility simulation comprises several critical components:

- **Traffic Models:** These define how vehicles and pedestrians move within the simulation, offering microscopic, macroscopic, or hybrid perspectives.
- **Road Network Representation:** Accurate representation of the road network, including road types, lanes, intersections, and constraints.
- **Vehicle Dynamics:** Parameters like acceleration, deceleration, and turning behavior are modeled to simulate realistic traffic.
- **Traffic Control Systems:** Various control systems such as traffic lights, stop signs, and pedestrian crossings are integrated into simulations.

Simulation methodologies include:

- **Agent-Based Simulation:** Modeling individual entities as autonomous agents, facilitating fine-grained analysis.
- **Microscopic Simulation:** Focusing on individual vehicle behaviors for detailed analysis.
- **Macroscopic Simulation:** Analyzing traffic flow at a higher level, treating vehicles as flow units.

5.1.1 SUMO

Simulation of Urban Mobility (SUMO)[15] is a widely used open-source traffic simulation software designed for modeling and simulating urban transportation systems. Developed in Python, SUMO provides a comprehensive framework for researchers, urban planners, and traffic engineers to analyze and optimize urban traffic flow.

SUMO allows users to create detailed and realistic simulations of urban road networks, including various traffic elements such as vehicles, pedestrians, traffic lights, and public transport. Its key features include:

- **Traffic Networks**: SUMO enables the creation of road networks with different types of intersections, lanes, and road geometries, providing a realistic representation of urban infrastructure.
- **Vehicle Models**: It supports a range of vehicle models, allowing for the simulation of various vehicle types, including cars, trucks, and bicycles, each with customizable behavior and characteristics.
- **Traffic Control**: SUMO allows for the implementation of advanced traffic control strategies, including traffic lights, stop signs, and priority rules. Researchers can experiment with different control algorithms to optimize traffic flow.
- **Public Transport**: The software can simulate public transportation systems, including buses, trams, and subways, making it valuable for studying multimodal transportation in urban areas.
- **Traffic Demand Generation**: Users can generate realistic traffic demand patterns, including origin-destination matrices, to model the movement of people and vehicles within the urban environment.

SUMO finds applications in various domains, including:

- **Traffic Management**: SUMO aids in evaluating and optimizing traffic management strategies, such as adaptive traffic signal control and congestion management.
- **Urban Planning**: It assists urban planners in assessing the impact of infrastructure changes and proposed transportation projects on traffic flow and congestion.
- **Research and Development**: Researchers use SUMO for developing and testing traffic control algorithms, autonomous vehicle systems, and intelligent transportation solutions.
- **Education**: SUMO serves as an educational tool for students and professionals interested in transportation engineering and urban mobility.

Simulation of Urban Mobility (SUMO)[15] plays a pivotal role in enhancing our understanding of urban traffic dynamics and optimizing traffic flow. Its flexibility, extensibility, and open-source nature make it a valuable resource for studying and improving urban mobility systems.

5.1.2 CityFlow

The widely used public traffic simulator, SUMO (Simulation of Urban Mobility)[15], has limitations in terms of scalability to accommodate large road networks and traffic flows. The authors mention that SUMO's performance deteriorates significantly when simulating extensive road networks and a high volume of vehicles, particularly when interfacing with Python for reinforcement learning support. In contrast, the authors introduce their novel traffic simulator, CityFlow[30], which addresses these limitations. CityFlow offers multithreading capabilities and is the first open-source simulator designed to support city-wide traffic simulation. It offers flexibility in defining road networks, vehicle models, and traffic signal plans, boasting a simulation speed over twenty times faster than SUMO. Additionally, the authors provide a user-friendly interface tailored for reinforcement learning experiments.

The implementation of CityFlow not only offers promise for enhancing traffic signal regulation, but it also provides doors for a variety of large-scale transportation research investigations, such as the avoidance of traffic jams and the routing of vehicles via mobile applications. In addition, CityFlow has the potential to act as a standard environment for reinforcement learning in the field of transportation research, very much like OpenAI Gym. In addition, the authors state that they want to improve the simulator by calibrating its simulation settings using observations from the actual world, which will result in the generation of data samples that are both quick and "real."

The purpose of this study is to investigate the capabilities and prospective uses of CityFlow, with a particular emphasis on CityFlow's contribution to tackling the issues of urban traffic control and improving transportation research through reinforcement learning.

5.1.3 Chosen Option: SUMO

When selecting a simulation tool for my research on optimizing urban traffic flow using reinforcement learning-based traffic light control, it was essential to consider the strengths and weaknesses of available options. In this section, we elaborate on my choice of Simulation of Urban Mobility (SUMO) over CityFlow, another prominent simulation testbed.

Zhang et al. (2019)[30] introduced CityFlow as a simulation testbed for urban traffic management. However, a critical examination of CityFlow revealed two significant drawbacks that influenced my decision:

One of the primary concerns with CityFlow, already briefly stated in Section 2.3, is the absence of rigorous calibration and evaluation within the general transportation community. In contrast, SUMO has been widely embraced and validated by transportation researchers and professionals. While CityFlow claims to produce equivalent output to SUMO, this assertion is primarily based on results from simplified grid network scenarios. These scenarios may not capture the complexity and nuances of real-world urban traffic dynamics. SUMO, on the other hand, benefits from extensive calibration and evaluation, making it a trusted tool in the transportation field.

CityFlow's common benchmark scenario, the Manhattan, NY network, is often cited as representing a real-world city layout and demand. However, the support for this claim is limited. In contrast, SUMO offers a rich array of benchmark scenarios, including those derived from actual urban environments, making it a more versatile choice for simulating real-world traffic conditions. This versatility aligns with my research goal of optimizing urban traffic flow, which requires realistic modeling and evaluation.

It was essential for me to choose a simulation tool that not only offers a reliable and well-validated framework, but also makes it possible to accurately simulate different urban traffic scenarios. This was necessary for my thesis, which focuses on improving the flow of urban traffic through the use of reinforcement learning-based traffic light regulation. For the purposes of my study, SUMO is the tool of choice because of its comprehensive calibration and assessment capabilities, as well as its support for a wide range of real-world scenarios.

5.2 NetEdit

NetEdit is a powerful network editing tool developed as a part of the SUMO (Simulation of Urban Mobility)[15] suite. SUMO is widely used in the field of traffic simulation and optimization, and NetEdit is a crucial component of this framework. This section provides an overview of NetEdit, its features, and its significance in the context of traffic network modeling.

For the sake of traffic simulation, NetEdit was developed to make the process of creating and modifying road networks as easy as possible. It provides a graphical interface that is intuitive and easy to use, making it possible for academics, urban planners, and traffic engineers to create, amend, and improve road networks with relative simplicity. This tool is a key component in the SUMO ecosystem, as it gives users the ability to alter network architectures, road geometry, traffic signal arrangements, and a great deal more.

5.2.1 Key Features and Functions

- **Network Creation**: NetEdit enables users to create road networks from scratch. Users can define road segments, intersections, lanes, and various road attributes to design a detailed and realistic network.
- **Import and Export**: NetEdit supports the import of existing network data from various formats, allowing users to work with real-world road network data. It also provides export capabilities to save the edited networks for use in SUMO simulations.
- **Traffic Light Configuration**: One of the standout features of NetEdit is its ability to configure traffic lights and control strategies. Users can define traffic light phases, timings, and synchronization to optimize traffic flow.
- **Geometry Editing**: NetEdit allows precise editing of road geometries, including the adjustment of road widths, lane markings, and turn lanes. This level of detail is crucial for accurately modeling traffic behavior.

- **Validation and Simulation Integration**: The tool includes validation features to check the integrity of the network design. Moreover, NetEdit seamlessly integrates with SUMO's traffic simulation capabilities, enabling users to visualize and evaluate traffic scenarios.

5.2.2 Role in Generating the "Vake" Map Network

The building of the "Vake" map network and the subsequent traffic simulations were both significantly aided by NetEdit. A notable case study in the optimization of urban traffic is represented by the "Vake" map. Researchers made use of NetEdit's capabilities in order to construct a comprehensive and accurate road network for the Vake area. This road network took into account real-world data as well as traffic patterns.

By using NetEdit, they were able to:

- Accurately model the road network layout in the Vake district, considering various road types and intersections.
- Configure traffic lights at critical junctions to simulate different traffic management strategies.
- Fine-tune road geometries and lane configurations to match the actual road infrastructure.

This detailed network, created and edited with NetEdit, served as the foundation for conducting traffic simulations and optimizing urban traffic flow within the Vake district.

In conclusion, NetEdit is an indispensable tool within the SUMO framework, enabling researchers to create, edit, and optimize road networks for traffic simulations. Its role in generating the "Vake" map network exemplifies its significance in the field of urban traffic flow optimization.

5.3 Deep Learning Frameworks

5.3.1 PyTorch

PyTorch is a popular open-source deep learning framework developed by Facebook's AI Research lab (FAIR). It has gained widespread adoption among researchers and practitioners due to its flexibility, dynamic computational graph, and robust support for neural network development [19].

PyTorch stands out for several key features:

- **Dynamic Computational Graph**: Unlike some other deep learning frameworks, PyTorch uses a dynamic computational graph. This means that the graph is built on-the-fly as operations are performed, allowing for dynamic and intuitive model development and debugging.
- **Automatic Differentiation**: PyTorch offers automatic differentiation through its `autograd` package, which simplifies the training of neural networks by automatically calculating gradients for backpropagation.
- **Wide Adoption**: PyTorch is widely adopted in both academia and industry, making it a valuable choice for research and production-level deep learning projects.
- **Rich Ecosystem**: The PyTorch ecosystem includes various libraries and tools like torchvision for computer vision, torchtext for natural language processing, and PyTorch Lightning for streamlined model training.

PyTorch has been applied to a wide range of machine learning and deep learning tasks, including:

- **Computer Vision**: PyTorch has been used extensively for image classification, object detection, image generation, and image segmentation tasks.
- **Natural Language Processing (NLP)**: Researchers and practitioners leverage PyTorch for tasks like text classification, machine translation, and sentiment analysis.
- **Reinforcement Learning**: PyTorch is a popular choice for developing and training reinforcement learning models, often in combination with libraries like OpenAI's Gym.

- **Scientific Computing**: PyTorch's flexibility extends to scientific computing, making it suitable for applications in fields like physics and biology.

PyTorch's ease of use, dynamic nature, and strong community support make it an excellent choice for AI and machine learning projects. It is particularly relevant to my research as we leverage PyTorch for developing and training reinforcement learning models for traffic light control.

5.3.2 TensorFlow

TensorFlow, developed by Google's Brain Team, is another prominent deep learning framework known for its scalability, flexibility, and extensive ecosystem. It has been widely adopted in academia and industry for a wide range of machine learning and deep learning tasks [1].

TensorFlow offers several distinctive features:

- **Static Computational Graph**: TensorFlow uses a static computational graph, which allows for advanced optimizations during model compilation and deployment. This can lead to improved performance in production environments.
- **TensorBoard**: TensorFlow includes TensorBoard, a powerful visualization tool that helps researchers and developers track and visualize the training process, model performance, and more.
- **Keras Integration**: TensorFlow provides a high-level API called Keras, which simplifies the development of deep learning models. It offers an easy-to-use interface for building neural networks.
- **Distributed Computing**: TensorFlow supports distributed computing, making it suitable for training large-scale deep learning models across multiple GPUs and machines.

TensorFlow has been applied to a wide array of machine learning tasks, including:

- **Computer Vision**: TensorFlow has been used for tasks such as image classification, object detection, and image generation. It is particularly well-suited for deploying models on mobile and embedded devices.
- **Natural Language Processing (NLP)**: Researchers and developers use TensorFlow for building and training models for machine translation, text generation, and sentiment analysis.
- **Reinforcement Learning**: TensorFlow is a popular choice for reinforcement learning research and applications, with support for various RL libraries like OpenAI's Gym.
- **Production Deployments**: TensorFlow's static graph compilation and support for serving models in production make it a preferred choice for scalable and high-performance applications.

5.3.3 Use Cases

PyTorch and TensorFlow are two well-known deep learning frameworks, and my thesis makes use of their capabilities in order to optimize urban traffic flow by using reinforcement learning to regulate traffic lights. Within the scope of my study, the formation and education of a wide variety of intelligent agents is made possible, in large part, thanks to the contributions of these frameworks.

PyTorch, which is well-known for its dynamic computational graph and broad support for the building of neural network architecture, is the basis for the majority of our various intelligent agents. PyTorch is utilized in the construction of the following agents, specifically:

- **MAXWAVE**: An agent designed to maximize the efficiency of wave-based traffic flow.
- **MAXPRESSURE**: Focused on optimizing traffic flow by minimizing traffic congestion and maximizing road usage efficiency.
- **IDQN**: Utilizing deep Q-networks to learn optimal traffic light control policies.
- **IPPO**: Employing Proximal Policy Optimization for traffic signal control.

- **MPLIGHT**: An agent designed for multi-phased traffic light control.
- **MPLIGHTFULL**: An extended version of MPLIGHT with additional functionalities.

The dynamic and flexible nature of PyTorch enables us to tailor these agents to specific traffic scenarios and experiment with different reinforcement learning approaches.

In parallel, we utilize TensorFlow, known for its scalability and static computational graph, to develop and train certain intelligent agents. Specifically, TensorFlow is employed for the following agents:

- **FMA2C**: A traffic light control agent based on Federated Multi-Agent Actor-Critic.
- **FMA2CFULL**: An extended version of FMA2C, enriched with additional functionalities and improvements.

When it comes to training these agents, the static graph compilation and support for remote computation that TensorFlow provides are extremely useful features. This is especially true in large-scale and performance-critical circumstances.

By combining PyTorch with TensorFlow, we are able to leverage the benefits of both frameworks to handle several facets of urban traffic flow optimization via reinforcement learning. This was made possible by combining the power of both frameworks. The foundation of my research technique is comprised of these frameworks as well as the agents that I have designed and developed myself.

5.4 Hardware Setup

Any computational experiment, but particularly those using intricate algorithms and simulations in the field of artificial intelligence, is highly dependent on the underlying hardware infrastructure in order to achieve a successful outcome. In this section, we present a comprehensive review of the hardware configuration that was used for my study, with an emphasis on both its strengths and its limitations.

5.4.1 Processor

The heart of our computational infrastructure is the processor. We have been conducting extensive experiments using an *Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz*. This quad-core processor, part of the Intel Core family, offers a base clock speed of 2.40GHz, which can be boosted to higher frequencies when required. While it provides a reliable computational foundation, it is essential to acknowledge that AI research often demands high computational power, and more advanced processors may offer improved performance.

5.4.2 Graphics Card

For tasks that involve heavy parallel processing and deep learning, a dedicated graphics processing unit (GPU) is instrumental. In our experiments, we have been relying on the *NVIDIA Corporation TU116M [GeForce GTX 1660 Ti Mobile]*. This GPU is known for its performance and ability to accelerate deep learning tasks. It provides support for CUDA (Compute Unified Device Architecture), making it suitable for various AI-related workloads. However, it's worth noting that more powerful GPUs are available, which can significantly enhance the speed of training and inference processes.

5.4.3 Continuous Experimentation

Our research journey has been marked by continuous experimentation, with our hardware setup running nonstop for three weeks, equivalent to 21 days. This extended period of operation was necessitated by the computational demands of our experiments and the complexity of the reinforcement learning-based traffic light control models we have been developing.

It is essential to point out that a financial limitation was reflected in the hardware that was utilized, more precisely my own personal laptop which was outfitted with the CPU and GPU described above. Although these components have shown that they are capable, serious research in artificial intelligence

frequently requires access to high-performance computer clusters or specialized gear that was developed specifically for deep learning applications. In spite of these constraints, the hard work and devotion of our team has allowed us to make substantial headway in improving urban traffic flow through the use of approaches based on reinforcement learning.

In conclusion, our hardware configuration, which consists of an Intel Core i5-9300H central processing unit and an NVIDIA GeForce GTX 1660 Ti Mobile graphics processing unit, has been the workhorse for our investigation, despite the restrictions that are inherently present. Even though we are limited by the resources available on our own hardware, we have been conducting consistent tests over the course of three weeks to demonstrate our dedication to advance the subject of urban traffic flow optimization.

6. Methodology

In this section, we will delve into the implementation aspect of the project. Initially, we will provide insights into the development of the Vake Map. Following that, we will elaborate on our modifications to integrate the RESCO repository into our specific use case. Subsequently, we will proceed to examine the system's evaluation, emphasizing its utilization of RESCO once more.

6.1 Why Vake Street

In the pursuit of optimizing urban traffic flow, it is essential to begin by selecting a target street that encapsulates the complexity and challenges of urban traffic management. In this section, we delve into the rationale behind our choice of Vake Street as the focal point of our thesis.

6.1.1 Significance of Vake Street

First and foremost, Vake Street stands out as one of the most bustling thoroughfares in Tbilisi, the capital of Georgia. The significance of this street arises from multiple factors that converge to create a traffic scenario of paramount interest for our study.

Academic Hub

Vake Street is home to no fewer than eight universities, making it an academic hub of considerable proportions. The presence of numerous educational institutions along this stretch of road naturally attracts a substantial volume of students, faculty, and staff. This academic concentration contributes significantly to the street's vibrant and dynamic atmosphere.

Business Epicenter

Beyond its academic allure, Vake Street also boasts a reputation as a prime location for business enterprises. It hosts a multitude of offices, firms, and establishments, making it one of the preeminent commercial centers in Tbilisi. The clustering of business activities results in a constant influx of commuters and visitors, further intensifying the street's traffic dynamics.

Residential Nexus

In addition to its academic and commercial prominence, Vake Street encompasses a sizable residential area. The coexistence of educational institutions, businesses, and residential zones within the same vicinity engenders a unique traffic ecosystem. The daily routines of residents, including commuting and daily errands, contribute significantly to the overall traffic load.

6.1.2 Traffic Challenges

The confluence of these factors underscores the challenges presented by Vake Street's traffic management. The street's infrastructure consists mainly of narrow lanes, typically featuring only one or two lanes on each side, alongside a dedicated bus lane. Notably, the introduction of bus lanes in recent times has added an extra layer of complexity to the traffic dynamics.

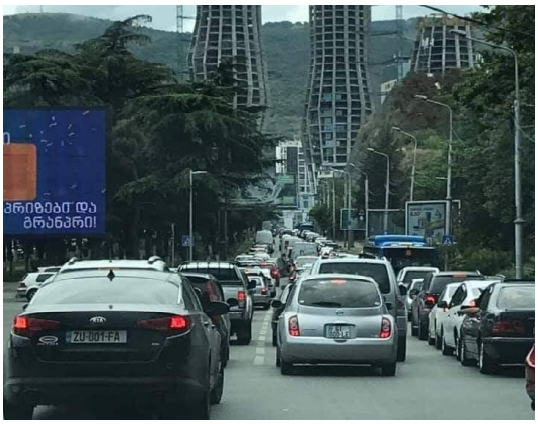


Figure 5: Typical Traffic Scene on Vake Street

Figure 5 provides a visual glimpse into the daily traffic scenario on Vake Street. It showcases the challenges posed by congestion, narrow roadways, and the constant interplay between vehicles, pedestrians, and public transportation.

6.1.3 Personal Connection

To add a personal dimension to our choice of Vake Street, it is worth mentioning that I, too, have had a direct experience with this street. As a former student of one of the universities located on Vake Street, I have personally witnessed the intricacies and frustrations of navigating this busy thoroughfare. My firsthand encounters with the traffic issues on Vake Street further fueled my determination to tackle this complex urban traffic optimization problem.

In summary, Vake Street's confluence of academic institutions, business enterprises, and residential communities, coupled with its traffic challenges and my personal connection, make it an ideal candidate for our thesis on optimizing urban traffic flow.

6.2 Creation of Vake Map

Before embarking on the simulation of urban traffic flow, it is imperative to have a map that closely mirrors the real-world environment. This level of realism is crucial for the accuracy and effectiveness of the simulation. In this section, we detail the process of creating the Vake map, which serves as the foundation for our traffic simulation.

6.2.1 Utilizing Netedit

To initiate the map creation process, we leveraged the powerful tool known as Netedit (Section 5.2). This tool is instrumental in designing road networks for traffic simulations. However, the challenge we encountered was the need for an exceptionally precise representation of the map. Manual creation proved to be a laborious and time-consuming endeavor.

6.2.2 Discovery of OpenStreetMap

Fortunately, our quest for an accurate map led us to the discovery of an invaluable resource - the OpenStreetMap (OSM) platform. OpenStreetMap provides comprehensive and authentic maps, including detailed information about various regions, including the prominent streets of Vake in Georgia.

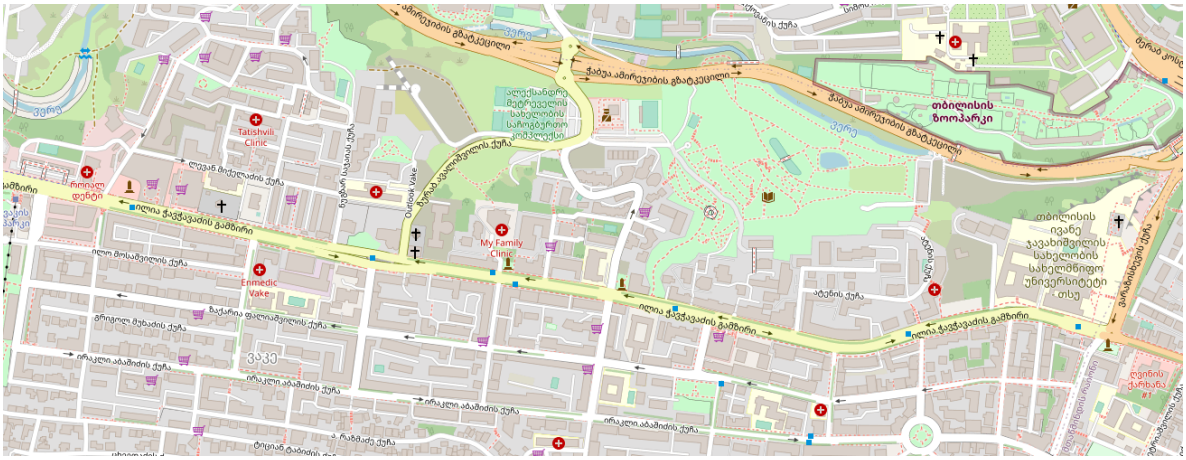


Figure 6: Vake Map from OpenStreetMap

As depicted in Figure 6, OpenStreetMap offers a user-friendly interface that allows us to select specific areas and generate map files with the extension *.osm*.

6.2.3 Conversion with *netconvert*

Once we obtained the OSM file, the next crucial step was the conversion of the map into a format compatible with the SUMO simulation environment. This conversion is accomplished using the SUMO tool called *netconvert*.

However, it's worth noting that the process doesn't conclude here. OpenStreetMap provides a comprehensive map that includes numerous objects and details that are not relevant for our SUMO simulation. Moreover, *netconvert* may introduce errors during the conversion process.

6.2.4 Refinement and Corrections

To ensure the accuracy and relevance of our map for traffic simulations, we embarked on a meticulous process of refinement and correction. This involved the removal of extraneous objects from the map and the addition of specific details required for our simulation, such as bus lines that were not initially present in the OpenStreetMap data.

The effort invested in this refinement phase exceeded our initial expectations, primarily due to the need for extensive object removal and the inclusion of additional details.

Figure 7: Vake Map for SUMO Simulation

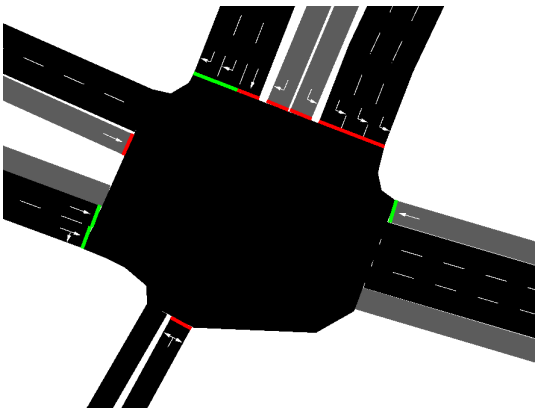
As seen in Figure 7, the result of our painstaking efforts is a map that faithfully replicates the shape and characteristics of the Vake area. This refined map serves as the groundwork for our subsequent traffic light control optimization experiments.

6.3 Traffic Lights

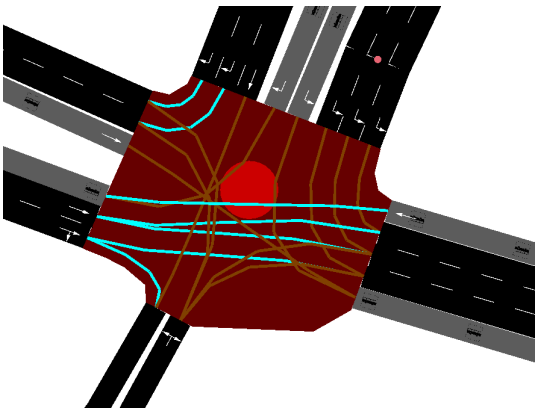
Traffic lights play a pivotal role in regulating and controlling traffic flow on urban streets. In this section, we will delve into the intricate world of traffic lights, which serve as both the cause and the tool for managing traffic in the complex environment of Vake Street. We will provide an in-depth examination of each of the six traffic lights situated along this thoroughfare.

6.3.1 Traffic Light 1

To commence our exploration of traffic lights, let's begin with an overview of Traffic Light 1, as depicted below:



(a) Intersection Overview



(b) Lane Directions

Figure 8: Traffic Light 1 (TL-1)

In Figure 8, we are presented with a comprehensive view of the complexity of this particular intersection. It's worth noting that this intersection stands out as one of the most challenging along

Vake Street, primarily due to the intricate array of lane directions and the presence of bus lines.

Intersection Complexity

The intersection depicted in Figure 8 exhibits a high level of complexity, characterized by the convergence of multiple road lanes and bus lines. Notably, this intersection comprises a myriad of lane directions, making it a focal point of traffic management concern.

Traffic Light Phases

Due to the inherent complexity of this intersection, Traffic Light 1 operates with a larger number of traffic light phases compared to other traffic lights along Vake Street. The additional phases are necessary to accommodate the diverse traffic movements, including the merging of bus lines and road lanes with non-standard directions.

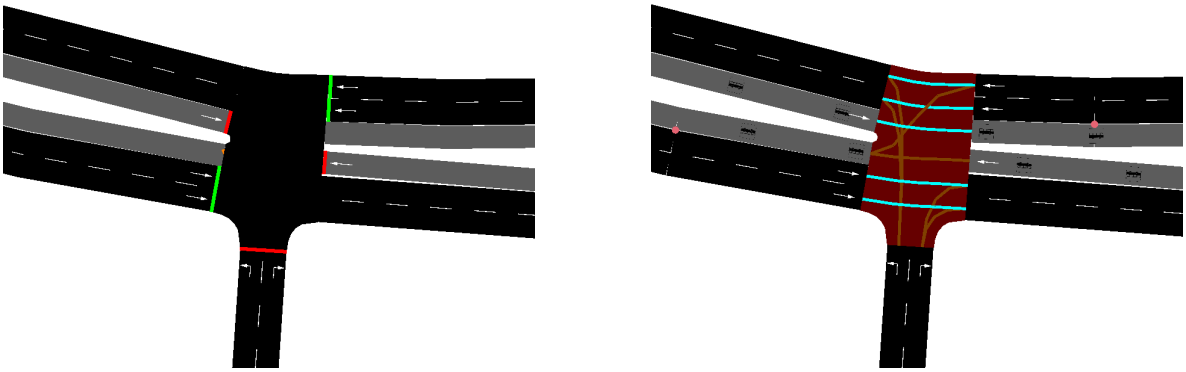
Bus Lane Configuration

Adding to the intricacy of this intersection is the configuration of bus lanes. These bus lanes run in a reverse direction compared to the adjacent vehicle lanes. Specifically, while the vehicle lanes may be oriented from east to west, the bus lanes run from west to east. As a result, within a total of six lanes, the top two lanes serve as vehicle lanes heading east, followed by one westbound bus lane, another eastbound bus lane, and finally, two more vehicle lanes heading west. This configuration adds an extra layer of complexity to both the road layout and the corresponding traffic light control.

In summary, Traffic Light 1 at this intricate intersection serves as a prime example of the challenges presented by the unique traffic dynamics of Vake Street. The presence of bus lanes, unconventional lane directions, and a multitude of phases underscores the significance of a thorough and nuanced analysis of traffic light management in our optimization efforts.

6.3.2 Traffic Light 2

Let's continue our exploration of traffic lights with an examination of Traffic Light 2, as illustrated below:



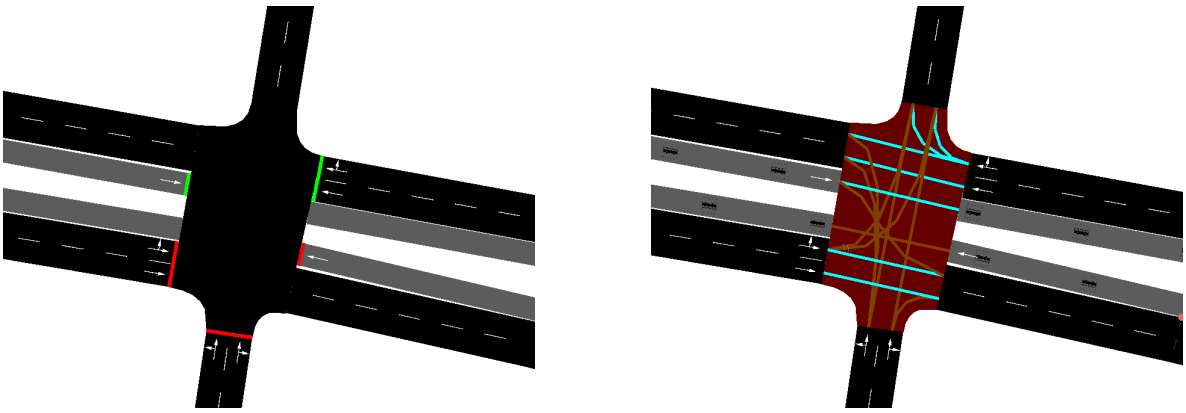
(a) Intersection Overview (b) Lane Directions

Figure 9: Traffic Light 2 (TL-2)

Comparing Traffic Light 2 in Figure 9 to Traffic Light 1, we observe a relative reduction in complexity. Notably, there is an absence of lanes heading north, simplifying the lane configuration. However, the consistent configuration of bus lanes, as detailed in previous sections, remains a common feature among all traffic lights on Vake Street.

6.3.3 Traffic Light 3

Our journey through the various traffic lights brings us to Traffic Light 3, portrayed below:



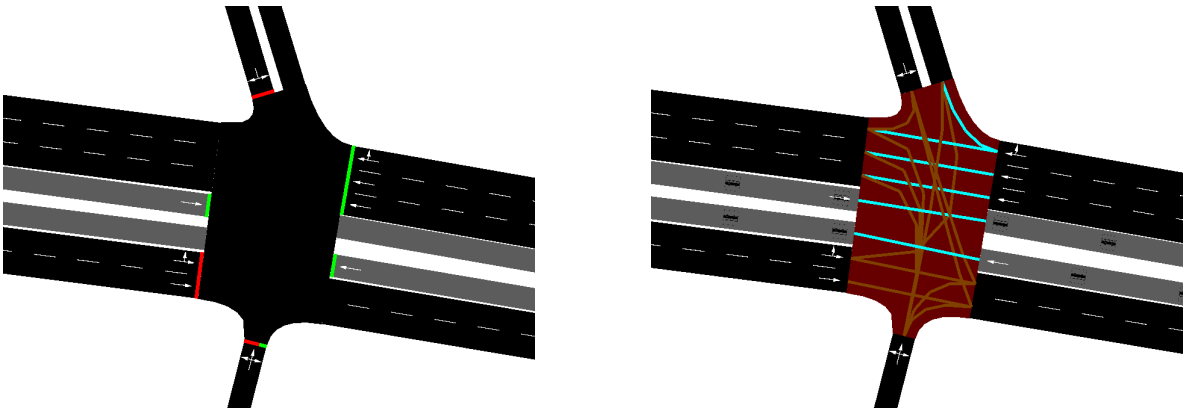
(a) Intersection Overview (b) Lane Directions

Figure 10: Traffic Light 3 (TL-3)

Traffic Light 3 introduces a heightened level of complexity compared to Traffic Light 2. This complexity arises from lanes originating from the south that have the potential to traverse east and west. Additionally, it's crucial to note that southbound turns are restricted at this intersection.

6.3.4 Traffic Light 4

Our exploration of traffic lights continues with Traffic Light 4, presented below:



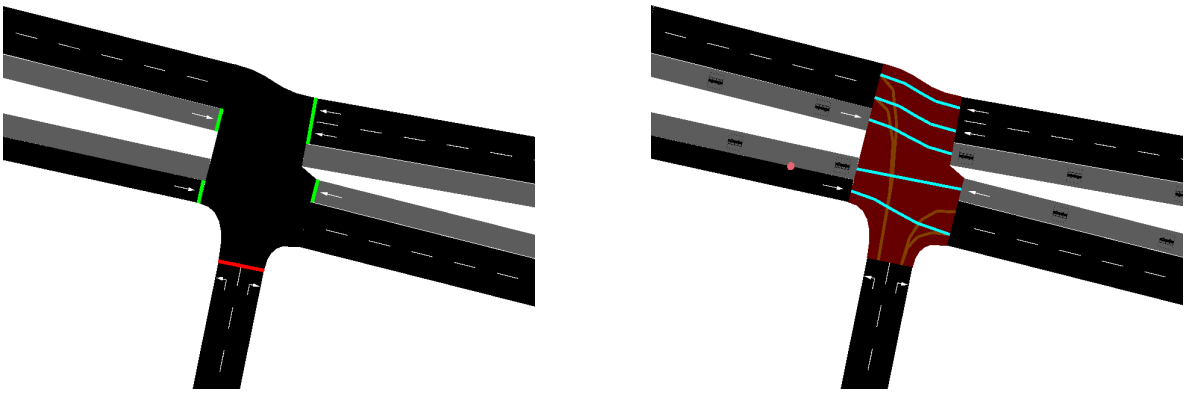
(a) Intersection Overview (b) Lane Directions

Figure 11: Traffic Light 4 (TL-4)

Traffic Light 4 presents a more complex scenario than Traffic Light 3. This intersection features additional directions and a reduction in the number of lanes in certain directions. For instance, traffic from the north can now turn in both east and west directions. Furthermore, we now have three lanes dedicated to eastbound traffic, further enhancing the lane dynamics.

6.3.5 Traffic Light 5

Our exploration reaches Traffic Light 5, depicted below:



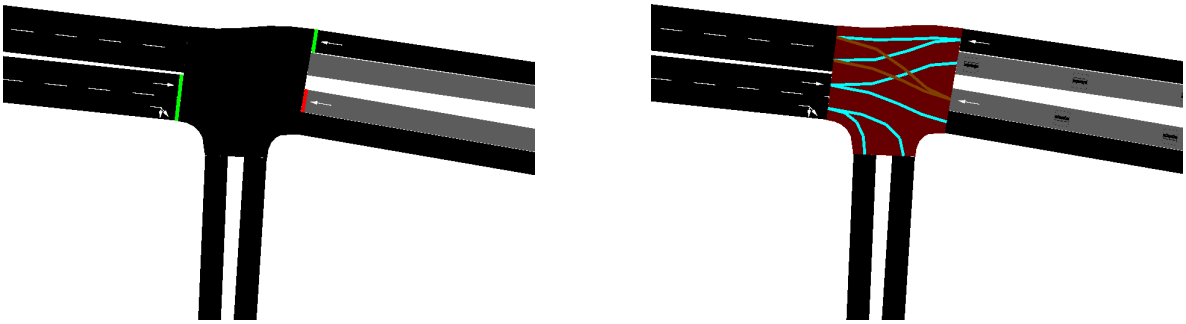
(a) Intersection Overview (b) Lane Directions

Figure 12: Traffic Light 5 (TL-5)

As we approach the end of Vake Street, Traffic Light 5 ushers in a simpler intersection configuration. The traffic light phases here are relatively straightforward, primarily handling traffic from the south and enabling direct westbound and eastbound movements. The consistent presence of bus lanes persists in this intersection, as with the previous ones.

6.3.6 Traffic Light 6

Our journey culminates with Traffic Light 6, the simplest intersection in this context, displayed below:



(a) Intersection Overview (b) Lane Directions

Figure 13: Traffic Light 6 (TL-6)

Traffic Light 6 represents the simplest intersection within Vake Street. Here, we primarily manage eastbound and westbound traffic, with the addition of southbound turns for westbound traffic. However, the essential role of traffic lights at this intersection is to facilitate the integration of bus lanes, which have reverse directions compared to vehicle lanes. The need for traffic lights arises from the necessity to harmonize these conflicting directions.

6.3.7 Conclusion

In the preceding sections, we have explored all six traffic lights along Vake Street. Each traffic light presents its own unique challenges, with varying levels of complexity. While some intersections

are relatively straightforward, others demand nuanced control due to the intricacies of lane directions and bus lane configurations.

As we delve further into our research, we will investigate the feasibility of employing independent and multi-agent systems for traffic light control. The potential of multi-agent systems lies in their capacity to create "Green Waves," allowing for consecutive green phases tailored to specific lanes and traffic demands. This exploration aligns with our overarching goal of optimizing urban traffic flow using reinforcement learning (RL) to intelligently determine green light allocations.

6.4 Traffic Simulation

In this section, we delve into the realm of traffic simulation. Having outlined the construction of the Vake map in preceding sections, it is imperative to simulate the traffic conditions on this map to represent the real-world challenges accurately. To achieve this, several crucial steps were undertaken.

First and foremost, to gain an in-depth understanding of traffic flow during the busiest hours of the day, which typically span from 5:00 PM to 9:00 PM, physical visits to the Vake Street location were conducted. These on-site visits, conducted over several days with intermittent intervals, provided valuable insights into the actual traffic patterns and directions during peak hours.

Subsequently, the knowledge gleaned from these field visits was translated into a digital simulation. This was accomplished with the assistance of SUMO's *randomTrips* tool. Despite its name, *randomTrips* is highly versatile, allowing us to specify a wide range of options, including probability distributions, to generate traffic scenarios tailored to our needs.

The process of generating an accurate simulation, however, was not without its challenges. It required rigorous efforts and persistence. After days of meticulous work and fine-tuning, we achieved a simulation that closely mirrors the real-world traffic dynamics on Vake Street. This simulation serves as the foundation for our experiments and analysis.

For the purpose of our experiments, we opted to run the simulation for a duration of 2.5 hours, focusing specifically on the busiest period of the day. This period has been carefully selected to capture the peak traffic conditions, allowing us to conduct a series of experiments and evaluations, as detailed in the subsequent chapters.

The traffic simulation serves as a pivotal component of our research, enabling us to test and evaluate various traffic light control strategies in a controlled and representative environment. It forms the basis upon which we conduct experiments to optimize urban traffic flow using reinforcement learning-based traffic light control.

6.5 Adaptation of RESCO

In this section, we will delve into the changes made to the public RESCO repository and its adaptation for use with our specific map of Vake Street.

To enable any map to function within the RESCO framework, we must first ensure that we have a network map of SUMO and a configuration file specifying the network and simulation we intend to use. As detailed in Sections 6.2, 6.3, and 6.4, we have already prepared these requisite files.

6.5.1 Map Config

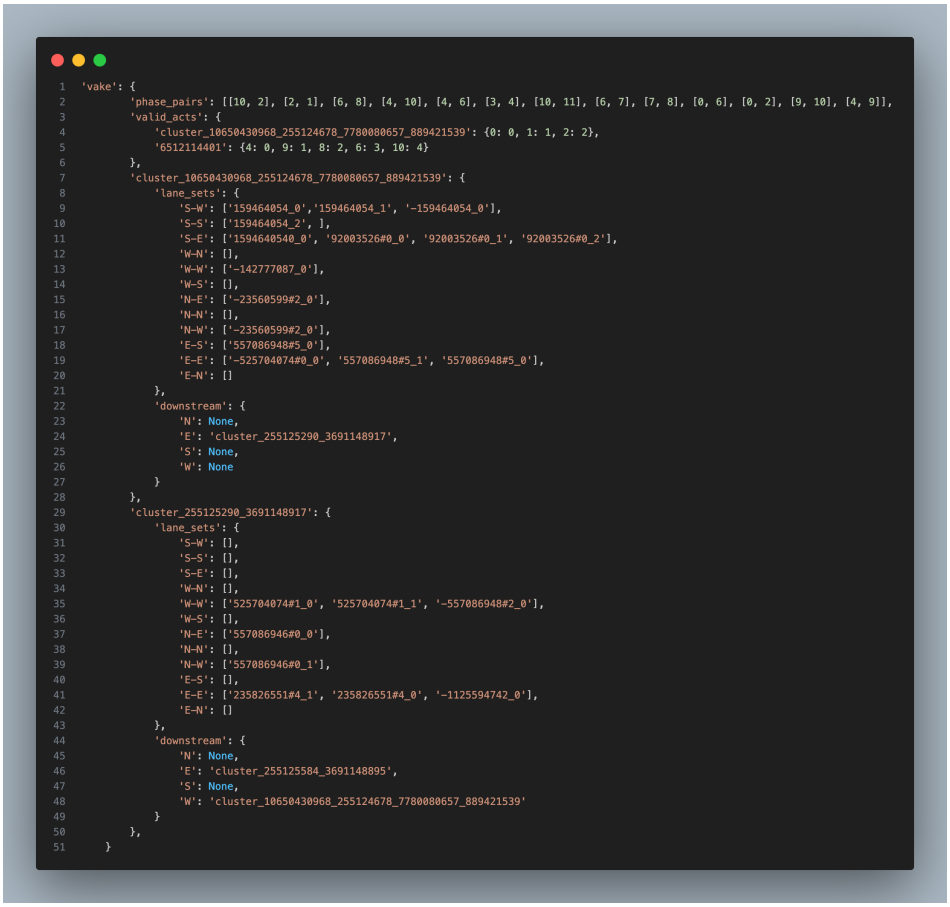
Once we have the necessary components in place, they must be integrated into the *map_config* of RESCO, which takes on the following format, as illustrated in Figure 14:

Figure 14: Map Config of Vake

Within the *map_config*, we specify crucial parameters, such as the fixed duration of the yellow phase, which remains constant and is not included in training due to its fixed time nature. Additionally, we define the warm-up period, indicating the number of seconds before the simulation begins to actively manage traffic signals.

6.5.2 Signal Config

Following the completion of the Map Config, we turn our attention to the Signal Config, one of the most critical configurations within the RESCO repository. Given the complexity of this configuration for multiple traffic lights, we will focus on the code snippet as shown below:



```
'vake': {
        'phase_pairs': [[10, 2], [2, 1], [6, 8], [4, 10], [4, 6], [3, 4], [10, 11], [6, 7], [7, 8], [0, 6], [0, 2], [9, 10], [4, 9]],
        'valid_acts': {
            'cluster_10650430968_255124678_7780080657_889421539': {0: 0, 1: 1, 2: 2},
            '6512114401': {4: 0, 9: 1, 8: 2, 6: 3, 10: 4}
        },
        'cluster_10650430968_255124678_7780080657_889421539': {
            'lane_sets': {
                'S-W': ['159464054_0','159464054_1', '-159464054_0'],
                'S-S': ['159464054_2', ],
                'S-E': ['1594640540_0', '92003526#0_0', '92003526#0_1', '92003526#0_2'],
                'W-N': [],
                'W-W': ['-142777087_0'],
                'W-S': [],
                'N-E': ['-23560599#2_0'],
                'N-N': [],
                'N-W': ['-23560599#2_0'],
                'E-S': ['557086948#5_0'],
                'E-E': ['-525704074#0_0', '557086948#5_1', '557086948#5_0'],
                'E-N': []
            },
            'downstream': {
                'N': None,
                'E': 'cluster_255125290_3691148917',
                'S': None,
                'W': None
            }
        },
        'cluster_255125290_3691148917': {
            'lane_sets': {
                'S-W': [],
                'S-S': [],
                'S-E': [],
                'W-N': [],
                'W-W': ['525704074#1_0', '525704074#1_1', '-557086948#2_0'],
                'W-S': [],
                'N-E': ['557086946#0_0'],
                'N-N': [],
                'N-W': ['557086946#0_1'],
                'E-S': [],
                'E-E': ['235826551#4_1', '235826551#4_0', '-1125594742_0'],
                'E-N': []
            },
            'downstream': {
                'N': None,
                'E': 'cluster_255125584_3691148895',
                'S': None,
                'W': 'cluster_10650430968_255124678_7780080657_889421539'
            }
        },
    }
```

Figure 15: Signal Config of Vake

The Signal Config, as depicted in Figure 15, is structured as a Python dictionary, closely resembling the JSON format. This configuration encompasses various fields, including *phase_pairs* and *valid_acts*. While not all agents utilize these fields, they are essential for specifying phase pairs and valid acts, dictating which phase pairs are applicable for each traffic light and phase. The indexes in phase pairs correspond to the index mapping found in Figure 16. Each phase pair represents a combination of traffic flow directions, and their indexing defines which pairs of traffic flows can be enabled together.

The Signal Config further includes fields for each traffic light, represented by their IDs in SUMO. These fields encompass *lane_sets* and *downstream*. In the *lane_sets* section, we list the IDs of individual lanes in the SUMO simulation and specify the traffic flows associated with each lane. Multiple traffic flows can be managed by a single lane, allowing for flexibility in modeling traffic patterns. The *downstream* section identifies adjacent traffic lights, which is crucial for multi-agent coordination. For instance, if Traffic Light 1 (TL-1) has Traffic Light 2 (TL-2) as its downstream neighbor in the eastward direction, TL-2 reciprocally recognizes TL-1 as its downstream neighbor to the west.

The configuration of the Signal Config posed significant challenges, mainly due to the lack of comprehensive documentation and explanations regarding its setup. Despite opening issues in the RESCO repository, obtaining guidance proved elusive, necessitating thorough investigation and experimentation to comprehend the intricacies of the configurations.

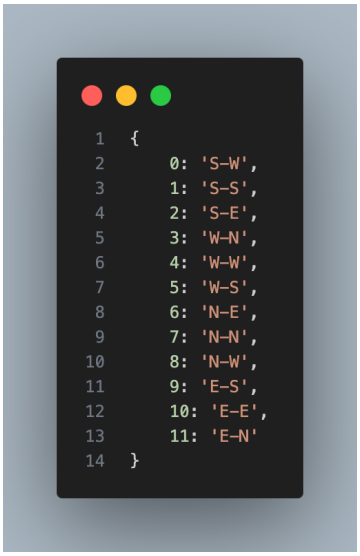


Figure 16: Indexes of Traffic Flows

6.5.3 MDP Config

Additionally, RESCO employs an MDP Config, which is primarily utilized by the FMA2C agent for Multi-Agent Traffic Control. The MDP Config structure includes various fields, many of which pertain to hyperparameters. However, two noteworthy fields are *management* and *management_neighbours*.

Management

Within the *management* field, traffic lights are categorized into groups, with each agent assuming responsibility for a specific group. In our case, we have two agents, *top_mgr* and *bot_mgr*, each overseeing three traffic lights. These agents interact with one another as part of the Multi-Agent Control framework.

Management Neighbors

The *management_neighbours* field specifies neighbors for each agent. While it appears relatively straightforward in our case due to the presence of just two agents, this configuration can accommodate more complex agent interactions and dependencies.

6.5.4 Contribution to RESCO

Throughout the course of this project, numerous challenges were encountered when working with RESCO, including issues with documentation and code errors. Some discrepancies in documentation and code had to be rectified to ensure successful execution. Consequently, efforts were made to address these issues, and in the spirit of open-source collaboration, a public Pull Request will be submitted to integrate the changes made to RESCO. This contribution aims to enhance the usability of RESCO for future researchers and students, mitigating some of the challenges encountered during this project.

6.6 Evaluation

In this section, we delineate the approaches we will employ to evaluate our work and the resultant findings.

One of the primary motivations behind the creation of RESCO was the absence of a general and publicly available benchmark for evaluating traffic light control systems. Consequently, RESCO offers a set of four key benchmark metrics for evaluation:

1. **Average Delay**: This metric quantifies the delay experienced by vehicles, measured as the time deviation from the expected travel time at the allowed speed. For instance, if a vehicle is permitted to travel at 60 km/h within a zone but is moving at 20 km/h due to traffic, the time difference represents the delay.

2. **Average Wait**: It measures the time vehicles spend stationary at traffic lights when their speed is reduced to zero.

3. **Average Queue**: This metric calculates the number of vehicles in an intersection when their speed reaches zero, providing insights into congestion levels.

4. **Average Trip Time**: It reflects the total time a vehicle spends within the vicinity of a traffic light, from entering the zone to exiting it.

These benchmark metrics serve as standardized criteria to assess the performance of traffic light control systems, enabling objective comparisons.

However, relying solely on these metrics may not reveal whether the underlying problem of optimizing urban traffic flow has been effectively addressed or if system improvements have been achieved. To address this concern, we conduct a comparative evaluation of various reinforcement learning (RL) algorithms against Baseline Controllers. Most notably, we employ the "Fixed Time Controller," the conventional traffic light control system commonly implemented in our target region.

Our evaluation process involves running simulations with the Fixed Time Controller, collecting data based on the aforementioned benchmark metrics, and subsequently comparing these results with those obtained from simulations using different RL algorithms. This comparative analysis enables us to assess the effectiveness and efficiency of RL-based traffic light control strategies in improving urban traffic flow.

By employing this comprehensive evaluation approach, we aim to ascertain the impact of our work on optimizing traffic flow within the context of Vake Street. This assessment will provide valuable insights into the effectiveness of RL-based traffic light control systems and their potential to mitigate traffic congestion and enhance overall urban mobility.

6.7 States and Reward Representation

In this section, we delve into the details of the state and reward representations used across our controllers. Understanding these representations is vital as they form the foundation upon which our RL-based traffic light control strategies are built.

6.7.1 States

State 1

State 1 encompasses the following observable states:

1. Total approach
2. Total wait
3. Total queue
4. Total speed

State 2

State 2 represents a normalized version of State 1.

State 3

State 3 consists of a single observable attribute, namely queue length.

State 4

In State 4, we incorporate the following values:

1. Total queue
2. Normalized total wait
3. Total speed
4. Normalized total approach

State 5

State 5, known as "Wave," represents the sum of queue length and approach, providing insights into traffic wave dynamics.

State 6

State 6 is specifically tailored for the FMA2C agent and includes state parameters such as approach and queue, accommodating specific hyperparameters.

State 7

Expanding on State 6, State 7 introduces additional parameters to the state representation, encompassing total wait time, the number of vehicles, and speed.

6.7.2 Rewards

Reward 1

Reward 1 is defined as the negative total wait time, serving as a straightforward reward mechanism.

$$\text{Reward}(s) = -\sum_{\text{lane in signal}} \text{wait\_time}(s_{\text{lane}}) \tag{1}$$

Reward 2

Reward 2 is a normalized variant of Reward 1, providing scaled rewards.

Reward 3

Termed as "Pressure reward," Reward 3 is represented by the negative queue length, promoting efficient traffic flow.

$$\text{Reward}(s) = -\sum_{\text{lane in signal}} \text{queue\_length}(s_{\text{lane}}) \tag{2}$$

Reward 4

Reward 4 is designed specifically for FMA2C due to its multi-agent nature. It combines departures, arrivals, the number of vehicles, queue length, and maximum wait time. The formula used for calculating the reward is as follows:

$$\text{Reward}(s) = -\sum_{\text{lane in signal}} \text{queue}(s_{\text{lane}}) - \text{coef} \times \sum_{\text{lane in signal}} \text{max\_wait}(s_{\text{lane}}) \tag{3}$$

Where:

$$\begin{aligned}
\text{Reward}(s) &: \text{The total reward for the current state } s. \\
\text{lane} &: \text{An individual lane in the signal.} \\
\text{queue}(s_{\text{lane}}) &: \text{The queue length in lane } s_{\text{lane}}. \\
\text{coef} &: \text{A coefficient defined in the configuration.} \\
\text{max_wait}(s_{\text{lane}}) &: \text{The maximum wait time in lane } s_{\text{lane}}.
\end{aligned}$$

This formula represents the reward calculation in your code, where you sum the negative queue lengths and the product of the maximum wait times and the coefficient for each lane in the signal. The resulting reward is negative because you subtract it from zero.

Reward 5

Reward 5 is tailored for FMA2CFull, another multi-agent system. Similar to Reward 4 with just change of coefficients and its values

With a clear understanding of the states and reward representations, we can now proceed to examine each controller and their respective results.

7. Experiments

In this chapter, we embark on a journey of experimentation and evaluation. Our primary objective is to conduct a comprehensive assessment of the various controllers and reinforcement learning (RL) algorithms employed in our study. To achieve this we will present the results obtained from running these controllers through simulations. It's worth noting that each RL controller underwent two separate runs, each comprising 1500 episodes.

7.1 Baseline Controllers

7.1.1 Fixed Time

In the Fixed Time Controller, we employ a rudimentary approach where traffic light phases are predetermined and follow a fixed timing schedule. Unlike our RL-based controllers, this baseline controller operates without any state or reward representation because it adheres strictly to predetermined timing intervals. Consequently, there is no dynamic adjustment of traffic lights based on real-time traffic conditions.

The Fixed Time Controller's results provide us with a benchmark against which we can compare the performance of our RL-based controllers. These results encompass various metrics, including but not limited to average delay, average wait time, average queue length, and average trip time. These metrics are crucial for assessing the effectiveness of more sophisticated traffic light control strategies.

7.1.2 Stochastic

The Stochastic Controller is a straightforward baseline where traffic light phases are selected randomly without any intelligent decision-making process. This controller serves as a simple reference point to evaluate how well our RL-based models perform compared to random traffic light control.

The results from the Stochastic Controller experiment offer insights into the outcomes of randomly selecting traffic phases. By comparing these results with those of our RL-based controllers, we can assess whether our models provide more efficient traffic control than random decision-making.

7.1.3 Max Wave

In the Max Wave Controller, we leverage the insights provided by State 5, which quantifies the traffic wave in the area. This baseline controller aims to minimize traffic wave effects by prioritizing the traffic light phase corresponding to the most significant wave.

The results of the Max Wave Controller experiment help us gauge the effectiveness of using wave analysis to guide traffic light control decisions. By examining metrics such as average delay, wait times, and queue lengths, we can assess whether this approach mitigates traffic wave-related issues.

7.1.4 Max Pressure

The Max Pressure Controller relies on the information provided by State 3, which quantifies the pressure on the traffic intersection based on queue length. This baseline controller seeks to alleviate congestion by choosing traffic light phases that reduce queue lengths.

The results from the Max Pressure Controller experiment provide insights into the impact of queue length on traffic light control. Metrics such as average queue length, delay, and wait times are essential for evaluating whether this approach effectively minimizes congestion compared to other controllers.

With the assessment of these baseline controllers, we can establish a foundation for evaluating the performance of our more sophisticated RL-based traffic light control strategies.

7.1.5 Results

Baseline Controller Results

In this section, we delve into the outcomes of our baseline controllers, which serve as the foundational benchmarks for evaluating the performance of more advanced RL agents. The results are visually presented in Figure 17.

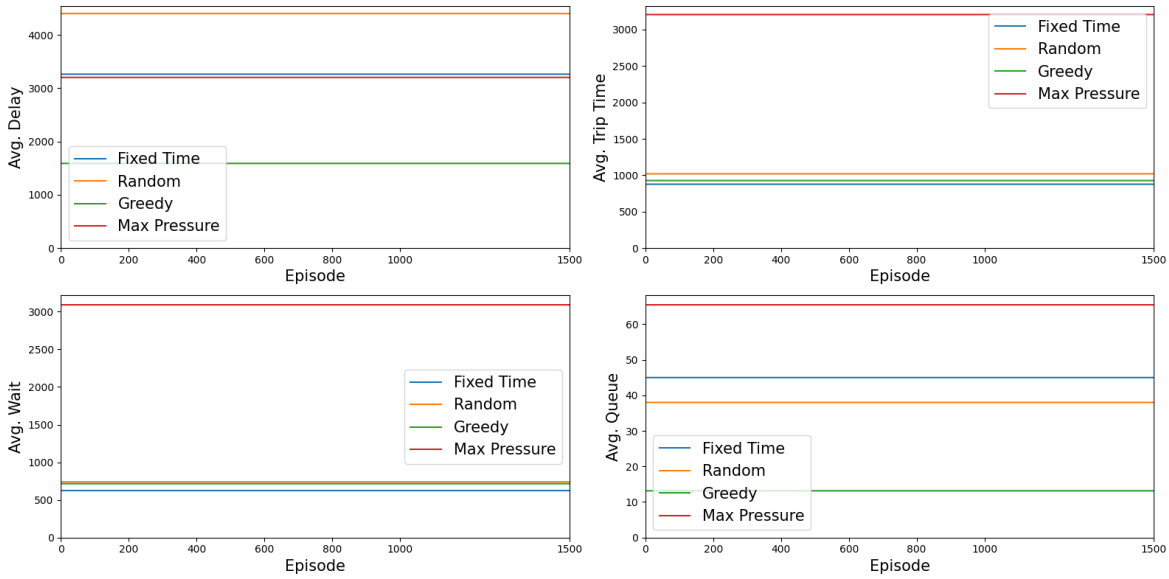


Figure 17: Baseline Results

These baseline controllers play a pivotal role in our evaluation process, as they set the fundamental performance standards against which RL agents will be compared. Among the baseline controllers, particular attention is given to the Fixed Time Controller, as it closely emulates the real-world conditions of the target street.

Surprisingly, in certain scenarios, the Fixed Time Controller outperforms even the Max Wave and Max Pressure Agents. This phenomenon can be attributed to the highly optimized nature of Fixed Time control, which aims to provide a generalized solution across various traffic conditions. The baseline results, thus, offer valuable insights into the inherent challenges and complexities of traffic light control.

The comparison between our RL agents and these baseline controllers will shed light on the efficacy of reinforcement learning approaches in optimizing urban traffic flow.

7.2 IDQN

The IDQN (Independent Deep Q-Network) agent is designed to utilize both state and reward representations to make informed traffic light control decisions. For state representation, we have chosen State 2, which encompasses critical attributes such as approach, total wait time, queue length, and total speed. This comprehensive state representation provides the agent with a wealth of observable information, facilitating intelligent decision-making.

Regarding the choice of reward, we have opted for Reward 2, which represents the normalized total wait time. By using this reward metric, the agent aims to minimize the cumulative waiting time of

vehicles at the intersection.

The observable range for traffic lights, which defines the distance at which they can detect approaching vehicles, is set at 200 meters.

The results section for the IDQN controller will provide an in-depth analysis of the controller's performance in terms of various metrics, including average delay, wait times, queue lengths, and trip times. These results will be essential for evaluating the effectiveness of the IDQN agent in optimizing traffic light control.

7.2.1 Results

Let's begin by examining the outcomes obtained from the IDQN RL Controller. The results are illustrated in Figure 18 below:

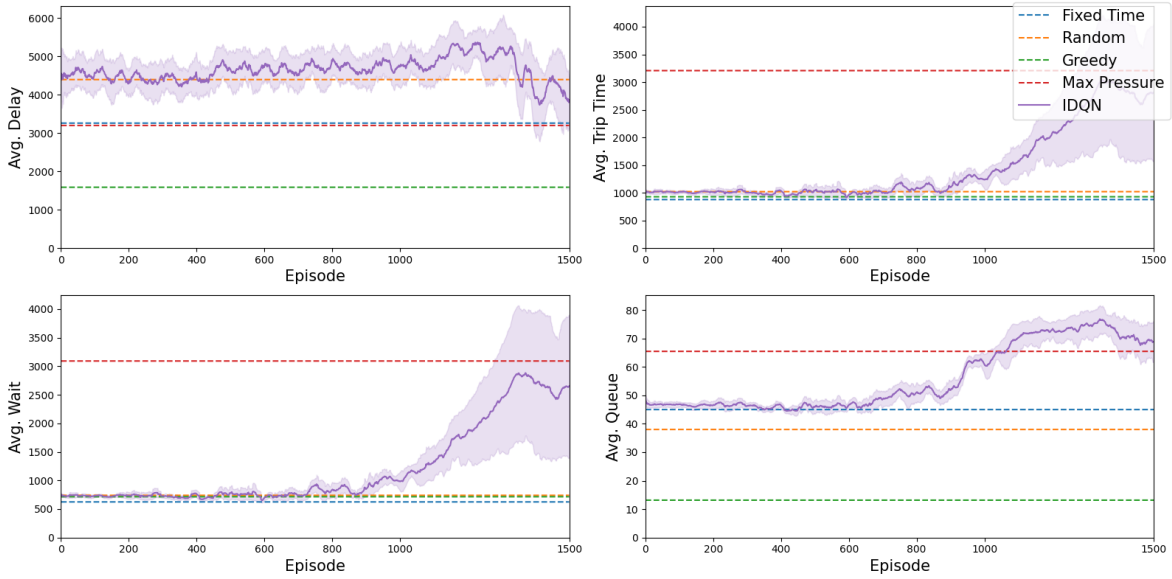


Figure 18: IDQN Performance Compared to Baselines

In Figure 18, we present the performance of the IDQN agent across four key metrics: Average Delay, Average Trip Time, Average Wait, and Average Queue. Analyzing these metrics in the context of this specific map, we observe that the IDQN agent did not outperform the Random Baseline Controller.

Considering that the reward function used for IDQN was Reward 2 and for state representation we used State 2, which is based on minimizing waiting time, it is noteworthy that IDQN initially demonstrated promising results. During the initial stages, it performed comparably to the Fixed and Greedy Baseline Controllers. However, as the simulation progressed beyond 1000 episodes, its performance deteriorated.

In summary, the IDQN RL Controller did not prove to be an optimal solution for the given map and traffic control problem. The observed decline in performance raises questions about the suitability of the selected state and reward representations for this specific scenario.

7.3 IPPO

The IPPO controller shares the same state and reward representations as the IDQN agent. It utilizes State 2, which encapsulates approach, total wait time, queue length, and total speed, and Reward 2, which normalizes the total wait time.

Similar to the IDQN agent, the maximum observable distance for traffic lights is set at 200 meters.

The results section for the IPPO controller will present an evaluation of its performance based on various metrics. These metrics will provide insights into the controller's ability to optimize traffic light control under real-world conditions.

7.3.1 Results

To begin, let's examine the results obtained from the IPPO RL Controller:

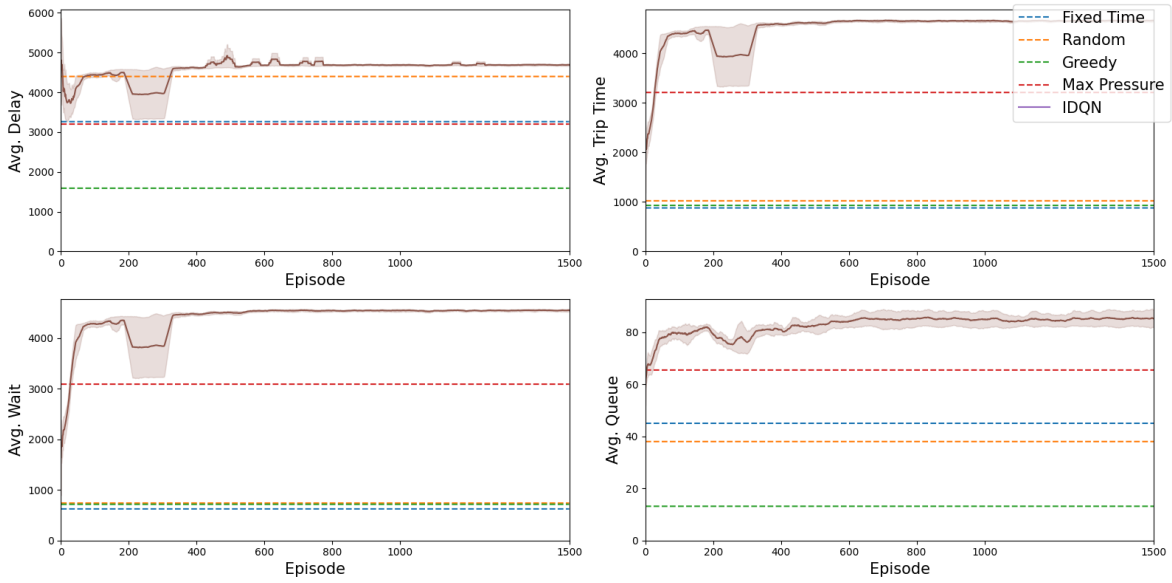


Figure 19: IPPO Performance Compared to Baselines

In Figure 19, we present the performance of the IPPO agent across four critical metrics: Average Delay, Average Trip Time, Average Wait, and Average Queue. Upon analyzing these metrics within the context of this specific map, it becomes evident that the IPPO controller did not perform favorably, failing to surpass any of the baseline controllers.

In contrast to the IDQN controller, which showed promise initially, the IPPO agent faced difficulties right from the start and consistently performed poorly throughout the simulation. The state representation employed by IPPO, State 2, encompasses crucial attributes such as approach, total wait time, queue length, and total speed. Additionally, IPPO utilized Reward 2, which normalizes the total wait time. However, even in the metric related to total wait time, IPPO exhibited subpar performance.

In summary, the IPPO RL Controller did not prove effective in managing the traffic flow within the given map. Its consistent underperformance across all metrics raises questions about its suitability for this specific traffic control scenario.

7.4 MPLight

The MPLight (Max Pressure with Deep Reinforcement Learning for Traffic Signal Control) controller adopts a different state and reward representation strategy compared to the previous agents. For state representation, we employ State 3, which focuses solely on the pressure at the intersection. This simplified state representation centers the agent's attention on the critical factor of pressure.

The reward chosen for MPLight is Reward 3, which is a negative value representing the queue length. This reward metric incentivizes the agent to minimize pressure and alleviate congestion.

Despite the change in state and reward representation, the observable range for traffic lights remains consistent at 200 meters.

The results section for the MPLight controller will provide an evaluation of its performance based on metrics related to queue length, delay, and other relevant factors. These results will help assess the effectiveness of MPLight in mitigating congestion at the intersection.

7.4.1 Results

Let's begin by examining the results obtained from the MPLight RL Controller:

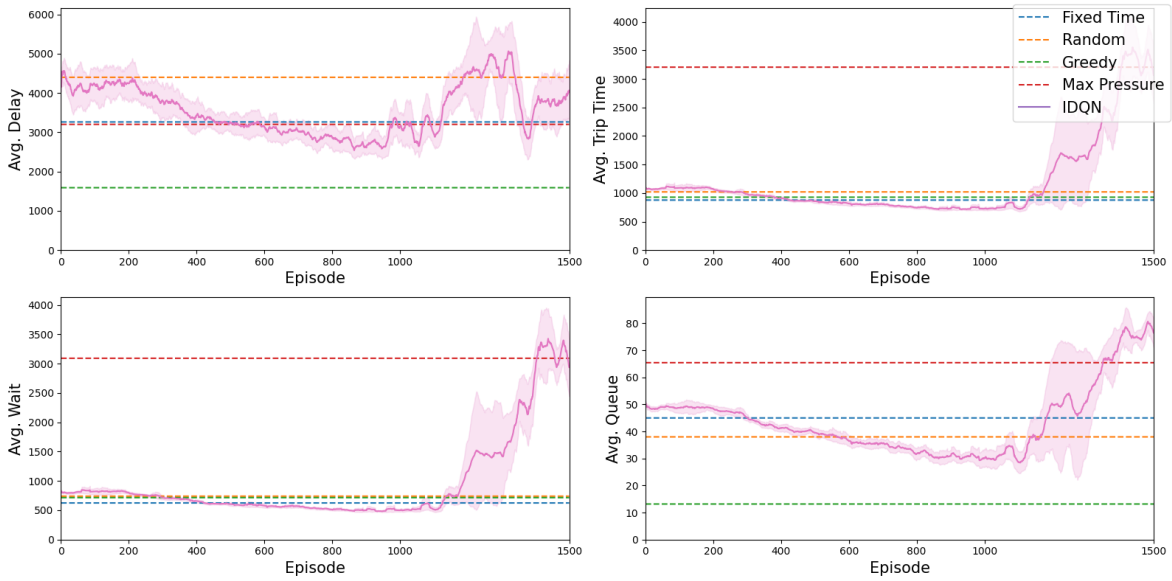


Figure 20: MPLight Performance Compared to Baselines

Figure 20 illustrates the performance of the MPLight agent across the same four essential metrics as the previous controllers: Average Delay, Average Trip Time, Average Wait, and Average Queue. Upon a detailed analysis of the results, it becomes evident that MPLight displayed a commendable performance overall.

In terms of Average Delay, MPLight managed to outperform all the baseline controllers, with the exception of the Greedy baseline, around the 800th episode. A similar trend can be observed in the Average Trip Time metric, where MPLight surpassed all controllers at approximately the same episode count. The story repeats itself when considering Average Wait Time, as MPLight excelled in performance around the same episode range.

Regarding Average Queue, MPLight reached its peak performance at around the 1250th episode, once again surpassing all baseline controllers except the Greedy baseline. Notably, MPLight relies on State 3 and Reward 3, which predominantly focuses on managing traffic pressure.

In summary, MPLight demonstrated an impressive ability to optimize traffic flow in the given scenario. While it may not have achieved significantly superior performance compared to baseline controllers, these results suggest that there is potential for further refinement and enhancement in its traffic control capabilities.

7.5 MPLightFull

MPLightFull is an extension of the MPLight controller. It employs a more comprehensive state representation, State 4, which includes queue length, normalized total wait time, total speed, and normalized approach. This richer state representation enables the agent to consider a broader range of factors when making traffic light control decisions.

The reward for MPLightFull remains the same as Reward 3, emphasizing the reduction of pressure as the primary objective.

Like its predecessor, MPLightFull operates with an observable range of 200 meters for traffic lights.

The results section for the MPLightFull controller will provide an evaluation of its performance based on the chosen state and reward representations. Metrics related to queue length, delay, and other relevant aspects will be analyzed to determine the controller's effectiveness in optimizing traffic flow.

7.5.1 Results

Let's begin by examining the results obtained from the MPLightFull RL Controller:

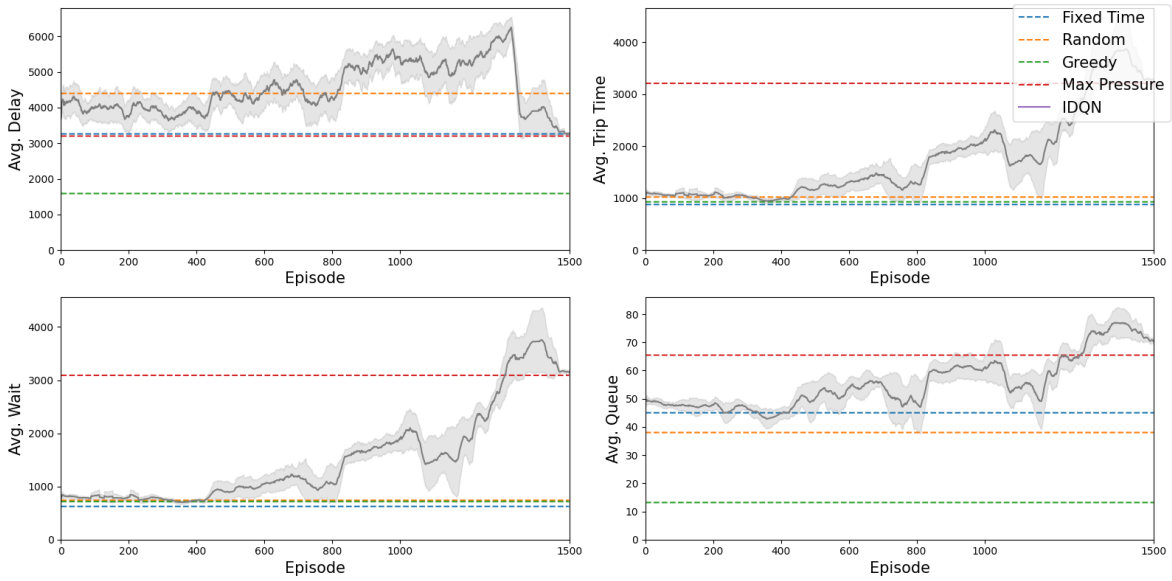


Figure 21: MPLightFull Performance Compared to Baselines

In Figure 21, we observe the results of the MPLightFull RL Controller across the same four fundamental metrics: Average Delay, Average Trip Time, Average Wait, and Average Queue, just as with the other controllers. It's intriguing to note that these results present an unexpected outcome.

Considering that the previous MPLight Controller Results displayed reasonably good performance, one might naturally expect that providing more extensive state representation through State 4 would yield even better results. However, the reality turned out to be quite the opposite. MPLightFull's performance, in fact, deteriorated in all metrics, and in some cases, it struggled to outperform the Random Baseline controller.

This outcome emphasizes a crucial point: an abundance of state representation and additional information does not necessarily translate into improved performance. MPLightFull's performance serves as a clear illustration of this phenomenon.

In summary, the MPLightFull RL Controller's results demonstrate that more extensive state representation alone does not guarantee superior performance and may even lead to unexpected outcomes.

7.6 FMA2C

The FMA2C controller is a multi-agent system that employs specific state and reward representations tailored to its collaborative nature. For state representation, FMA2C uses State 6, which is customized to meet the requirements of multi-agent traffic control. This state representation is primarily based on approach and queue length, essential for coordinating traffic light control among multiple agents.

The reward used in FMA2C is a combined reward metric found under Reward 4. This reward considers various factors, including departures, arrivals, the number of vehicles, queue length, and maximum wait time. It facilitates cooperative decision-making among agents.

The observable distance for traffic lights in FMA2C is set at 200 meters.

The results section for the FMA2C controller will provide a comprehensive evaluation of its performance in a multi-agent traffic control scenario. Metrics related to queue length, delay, and the collaborative behavior of the agents will be analyzed to assess the effectiveness of this multi-agent approach.

7.6.1 Results

Let's begin by examining the results obtained from the FMA2C RL Controller:

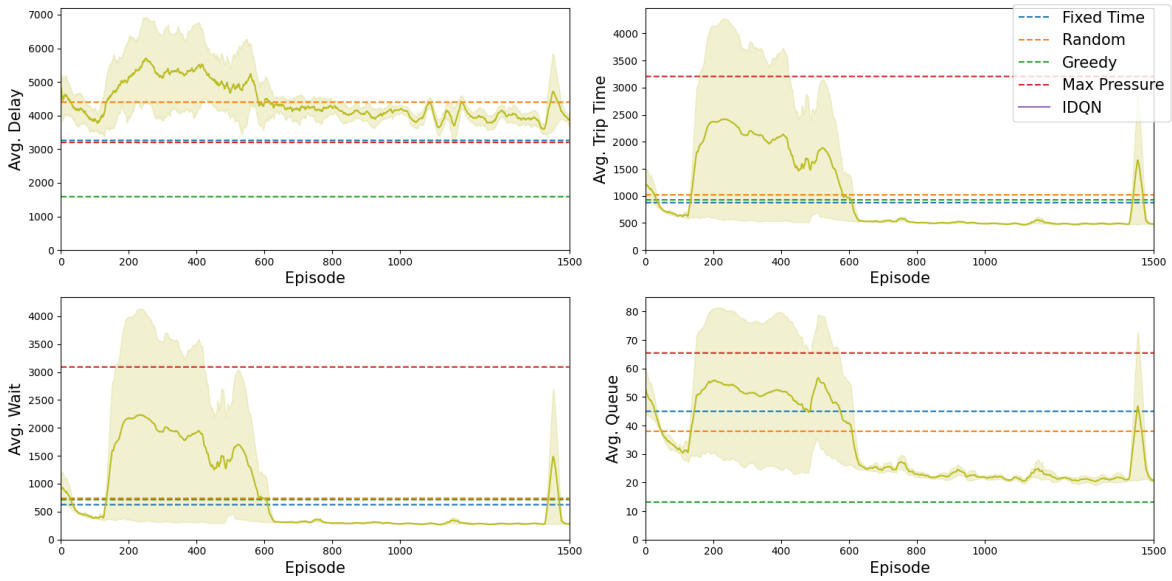


Figure 22: FMA2C Performance Compared to Baselines

In Figure 22, we can observe the results of the FMA2C RL Controller across the same four crucial metrics: Average Delay, Average Trip Time, Average Wait, and Average Queue, just as with the other controllers. It's important to note that the FMA2C Controller operates using a multi-agent control system.

When analyzing the Average Delay metric, we find that FMA2C did not excel during any phase of the training period. It mostly outperformed random controllers but struggled to compete with other baseline controllers. However, the standout performance of FMA2C is evident in both Average Trip Time and Average Wait Time metrics. These metrics share similarities due to their nature, and FMA2C demonstrated impressive results, surpassing all other controllers by a significant margin, with only a minor peak in performance around episode 500. Throughout training, it maintained a stable and noteworthy performance trajectory.

In the context of Average Queue, the pattern remains consistent, with a noticeable peak around episode 500 and overall stable progress. FMA2C significantly outperformed all other baseline controllers, with the exception of the greedy controller.

In summary, the FMA2C RL Controller exhibited remarkable performance results for this specific map and problem, particularly excelling in Average Trip Time and Average Wait Time metrics. These results underscore the effectiveness of the multi-agent control system employed by FMA2C in optimizing traffic flow.

7.7 FMA2CFull

FMA2CFull extends the capabilities of the FMA2C controller by incorporating a more comprehensive state representation, State 7. This state representation includes additional parameters such as total wait time, the number of vehicles, and speed, providing a more detailed view of the traffic conditions.

The reward for FMA2CFull remains consistent with Reward 5, which considers departures, arrivals, the number of vehicles, queue length, and maximum wait time.

The observable distance for traffic lights in FMA2CFull is set at 200 meters, aligning with the other controllers.

The results section for the FMA2CFull controller will evaluate its performance based on the extended state and reward representations. Metrics related to traffic flow, cooperative behavior among agents, and overall system efficiency will be analyzed to assess the advantages of this enhanced multi-agent approach.

7.7.1 Results

Let's begin by examining the results obtained from the FMA2CFull RL Controller:

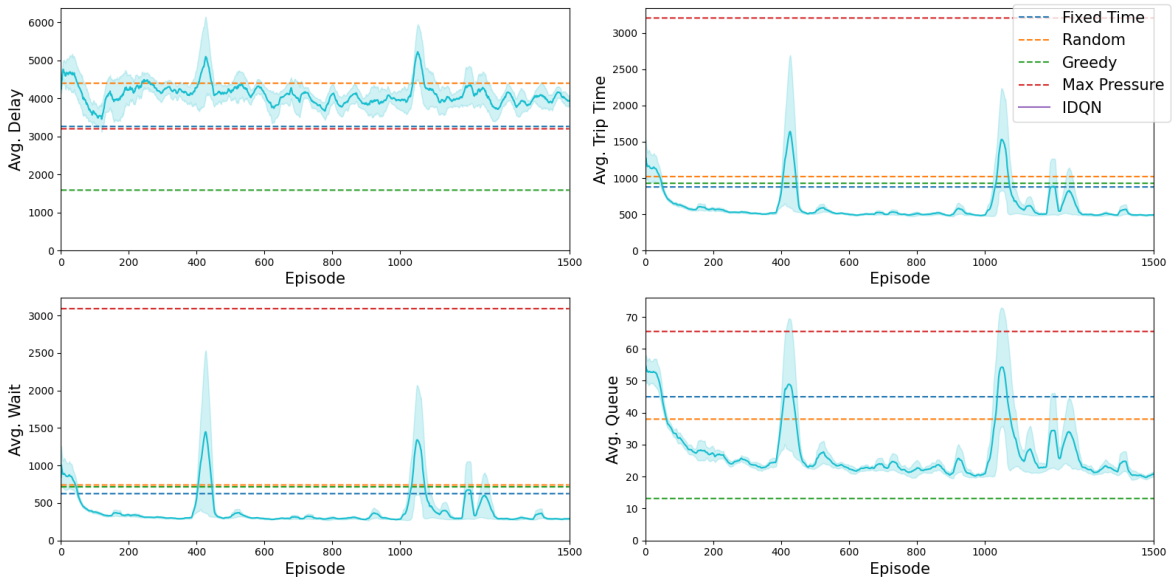


Figure 23: FMA2CFull Performance Compared to Baselines

In Figure 23, we can observe the results of the FMA2CFull RL Controller across the same four essential metrics: Average Delay, Average Trip Time, Average Wait, and Average Queue, consistent with the other controllers. Similar to the situations with MPLight and MPLightFull, this controller aims to increase the information available to the agent by altering the state representation and employing State 7.

In contrast to the previous case with MPLightFull, the extended state representation of FMA2CFull did not significantly deteriorate or enhance the agent's performance. It maintained a comparable performance level throughout the training process. Specifically, in the Average Delay metric, FMA2CFull exhibited similar performance to only the random controller. However, it notably outperformed other baseline controllers in both Average Wait and Average Trip Time metrics. In terms of the Average Queue metric, FMA2CFull significantly outperformed all baseline controllers, except for the Greedy controller. Notably, the performance curves and progress of FMA2CFull mirrored those of MPLight but with a peak occurring around episode 1000 rather than episode 500.

To summarize, the introduction of an extended state representation did not substantially alter the overall performance of FMA2CFull. It maintained a level of performance similar to its predecessor and continued to excel in Average Wait and Average Trip Time metrics while significantly outperforming baseline controllers in Average Queue.

7.8 Comparison

7.8.1 Controller Performance Comparison

In this section, we will provide a comprehensive comparison of the various controllers used in our experiments. To facilitate this comparison, we present the combined results of all controllers in Figure 24.

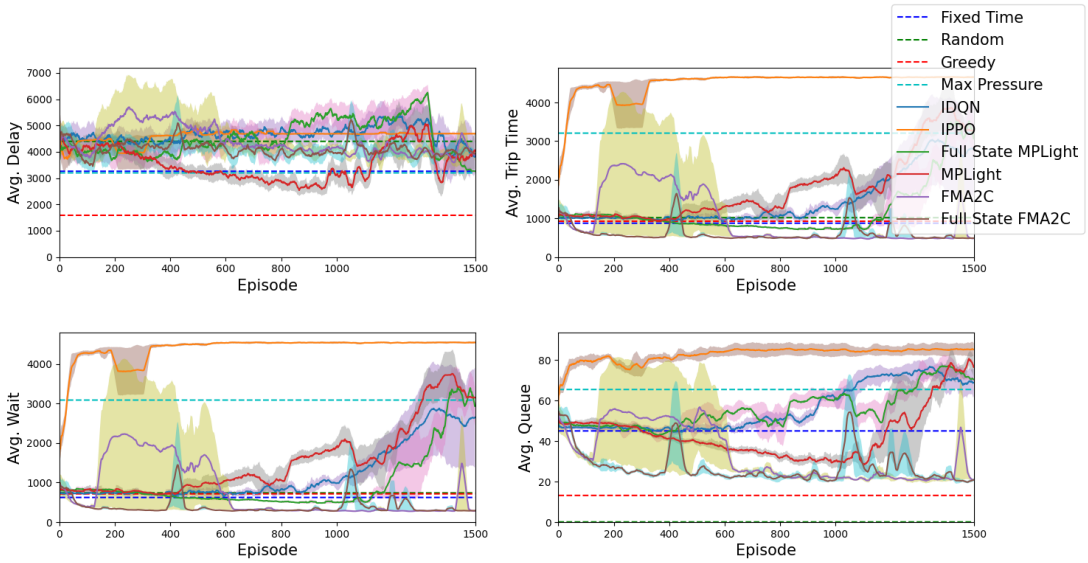


Figure 24: Combined Performance Results of All Controllers

Upon initial observation, Figure 24 may appear dense due to the simultaneous presence of multiple controllers. However, several noteworthy insights can be gleaned from this comprehensive overview. Let's delve into a few key observations:

1. **Average Delay:** In the context of Average Delay, MPLight stands out as a strong performer, surpassing all RL Controllers. Nevertheless, it falls short of outperforming the Greedy Controller, which excels in minimizing delay.

2. **Average Trip Time and Average Wait:** FMA2C and FMA2CFull controllers exhibit parallel performance trends in both Average Trip Time and Average Wait metrics. Following closely behind, MPLight maintains a competitive performance in these two crucial categories.

3. **Average Queue:** Similar to the Average Trip Time and Average Wait metrics, the performance of MPLight, FMA2C, and FMA2CFull aligns in terms of Average Queue management. These RL Controllers showcase impressive results compared to baseline controllers.

Overall, the comparative analysis of these controllers reveals nuanced performance patterns across various metrics. Each controller exhibits strengths in specific areas, highlighting the importance of selecting an appropriate controller based on the desired traffic management objective. The results underscore the potential of RL-based traffic light control methods to significantly improve urban traffic flow, with each controller contributing unique insights and performance nuances.

7.8.2 Performance Summary Tables

In this section, we provide a summary of the best performance achieved by each controller during training. Table 1 presents a comprehensive comparison of baseline and RL controllers across four key metrics: Average Delay, Average Trip Time, Average Wait, and Average Queue.

Table 1: Performance Comparison of Baseline and RL Controllers

Controller	Avg. Delay	Avg. Trip Time	Avg. Wait	Avg. Queue
Baseline Controllers				
FIXED	3262.59	**877.08**	**624.55**	-
STOCHASTIC	4399.18	1020.2	740.09	38.2
MAXWAVE	***1588.78***	927.81	715.55	***13.16***
MAXPRESSURE	3201.23	3206.6	3090.56	65.45
RL Controllers				
IDQN	3993.99	891.26	623.70	43.22
IPPO	3322.71	1183.85	896.15	51.80
MPLight	**2395.64**	705.73	480.66	22.09
MPLightFULL	3172.69	896.91	669.03	39.42
FMA2C	3475.60	***471.83***	***272.02***	**19.11**
FMA2CFULL	3345.40	477.61	277.39	19.24

In Table 1, we have categorized controllers into two groups: baseline and RL controllers. Within each category, the best-performing metric is bolded, and for the absolute best within that metric, we have used both bold and italic formatting.

For Average Delay, the MAXWAVE baseline controller exhibited the best performance, while among RL controllers, MPLight achieved the lowest delay. However, MAXWAVE outperformed MPLight in this metric, indicating that RL did not significantly improve this aspect.

In terms of Average Trip Time, the FIXED baseline controller and FMA2C RL controller displayed the best results, with FMA2C outperforming the others by almost twice the performance.

For Average Wait, the same pattern as Average Trip Time emerged, with FIXED and FMA2C leading the way. Here, FMA2C achieved more than double the performance of other controllers in the RL category.

In the Average Queue metric, MAXWAVE and FMA2C performed well, with MAXWAVE having a 50% lead over FMA2C.

Considering the overall results, the RL controller FMA2C consistently outperforms other RL controllers in three out of four metrics, making it a promising choice for this traffic management scenario.

7.8.3 Comparison with RESCO Results

In this section, we will compare our experimental results with those from RESCO ([4]). First, let's examine their findings presented in Figure 25.

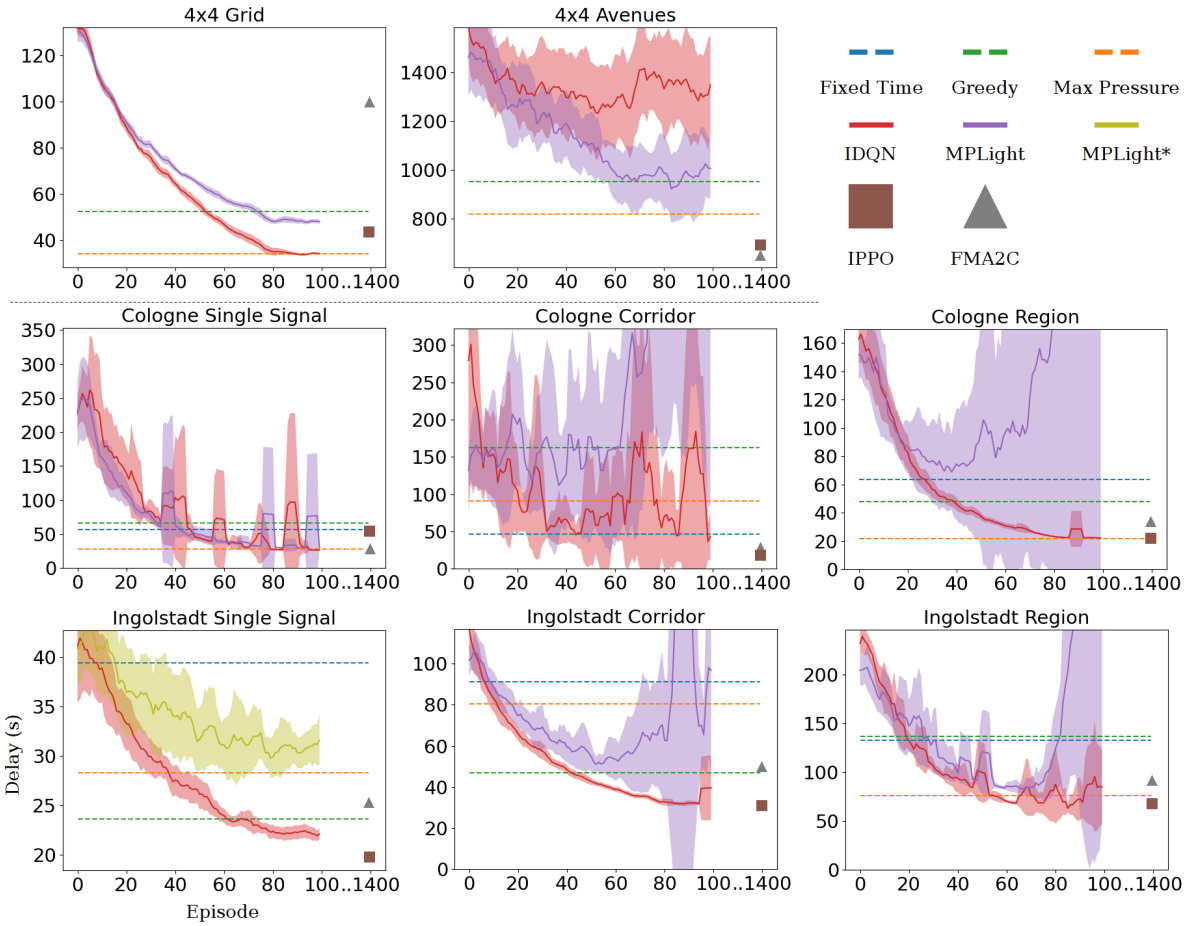


Figure 25: RESCO Results[4]

From RESCO's results, it is evident that IPPO and FMA2C performed exceptionally well. However, when we compare these results to our own experiments, the performance of IPPO was notably worse in our case, and only FMA2C demonstrated strong performance, consistently outperforming other controllers in our experiments.

It's crucial to note that RESCO's experiments encompassed various maps, and while their RL controllers excelled in almost all scenarios, the specific performance and progress can vary significantly depending on the map. This highlights the importance of tailoring RL models to the unique challenges presented by each map. Different maps may demand different strategies, and a one-size-fits-all approach is unlikely to yield optimal results.

In conclusion, our experiments align with RESCO's findings in terms of FMA2C's strong performance. However, the discrepancy in IPPO's performance underscores the need for map-specific research and the careful selection of RL models to address the distinct challenges posed by different maps.

8. Conclusion

In this research, we conducted a comprehensive evaluation of various traffic light control strategies, ranging from baseline controllers to advanced Reinforcement Learning (RL)-based agents, with the aim of optimizing urban traffic flow. Our study examined the performance of these controllers using a combination of state representations and reward metrics, yielding valuable insights into the effectiveness of different approaches.

8.1 Baseline Controllers

We initiated our exploration by studying baseline controllers, which served as critical reference points for evaluating the effectiveness of RL-based strategies. Four distinct baseline controllers were examined: Fixed Time, Stochastic, Max Wave, and Max Pressure.

The Fixed Time Controller represented a simplistic approach, employing predetermined traffic light phases without adaptation to real-time traffic conditions. Surprisingly, it even outperformed advanced RL-based controllers in specific scenarios, highlighting the challenges of achieving a generalized solution for traffic light control.

The Stochastic Controller, which randomly selected traffic light phases, provided a baseline against which we compared the RL-based models' performance, assessing whether RL models could surpass random control.

The Max Wave Controller aimed to minimize traffic wave effects by prioritizing the traffic light phase corresponding to the most significant wave, using State 5. The results indicated the potential of wave analysis in traffic control.

The Max Pressure Controller focused on alleviating congestion by selecting phases based on queue length, leveraging State 3. The results demonstrated the importance of queue length in traffic management.

8.1.1 Baseline Controller Results

The results of the baseline controllers set foundational benchmarks for evaluating RL agents' performance. While Fixed Time surprisingly outperformed some advanced controllers, these baseline results underscored the complexities and challenges inherent in traffic light control.

8.2 Reinforcement Learning-Based Controllers

Our evaluation included several RL-based controllers, each employing distinct state representations and reward functions to optimize traffic light control.

8.2.1 IDQN

The IDQN (Independent Deep Q-Network) agent, utilizing State 2 and Reward 2, initially showed promise in minimizing cumulative waiting time. However, its performance declined as training progressed.

8.2.2 IPPO

IPPO, utilizing the same State 2 and Reward 2 as IDQN, consistently underperformed baseline controllers, raising questions about its suitability for the given traffic control scenario.

8.2.3 MPLight

MPLight, focusing solely on traffic pressure using State 3 and Reward 3, displayed commendable performance. It outperformed baseline controllers in Average Delay, Average Trip Time, Average Wait, and Average Queue metrics, demonstrating its ability to mitigate congestion effectively.

8.2.4 MPLightFull

MPLightFull, an extension of MPLight with more extensive State 4 representation, exhibited unexpected results. Despite the increased state information, its performance deteriorated and struggled to outperform the Random Baseline controller, highlighting that additional state representation does not necessarily lead to improved performance.

8.2.5 FMA2C

FMA2C, a multi-agent system using State 6 and Reward 4, excelled in Average Trip Time and Average Wait Time metrics, surpassing all other controllers except the Greedy baseline. It demonstrated the effectiveness of cooperative multi-agent traffic control.

8.2.6 FMA2CFull

FMA2CFull, an extended version of FMA2C with State 7, maintained a performance level similar to its predecessor. It excelled in Average Wait and Average Trip Time metrics while significantly outperforming baseline controllers in Average Queue.

8.3 Final Insights and Implications

Our comprehensive evaluation provides valuable insights into traffic light control strategies. Each controller demonstrated strengths in specific areas, emphasizing the importance of selecting an appropriate controller based on the desired traffic management objective. The results underscore the potential of RL-based traffic light control methods to significantly improve urban traffic flow, with each controller contributing unique insights and performance nuances.

8.4 Comparison with RESCO Results

We also compared our experimental results with those from RESCO[4]. While our experiments aligned with RESCO's findings in terms of FMA2C's strong performance, discrepancies in IPPO's performance highlighted the need for map-specific research and the careful selection of RL models to address the distinct challenges posed by different maps.

In conclusion, this research sheds light on the complexities of urban traffic control and the potential of RL-based solutions. It encourages further exploration into tailored RL models and strategies for specific traffic scenarios, emphasizing the need for adaptive and context-aware traffic light control systems.

9. Future Work

In this chapter, we delve into the exciting realm of potential future endeavors that could further enhance the effectiveness of the research presented in this thesis. Numerous possibilities lie ahead, each holding the promise of enriching our understanding and application of traffic flow optimization.

9.1 Enhancing the Simulation Environment

One avenue for future work involves expanding the scope of our traffic simulation to create a more comprehensive and realistic environment. To achieve this, several key enhancements can be considered.

Firstly, the incorporation of bus simulations into the SUMO (Simulation of Urban MObility) environment stands as a pivotal step. The inclusion of buses introduces an additional layer of complexity to the traffic dynamics. Consequently, a natural progression involves adapting the reward functions to prioritize public transport vehicles, ensuring minimal delays for buses. This adjustment acknowledges the importance of efficient public transportation systems in urban areas and aligns with the broader goal of sustainable urban mobility.

9.2 Pedestrian Simulation Integration

Taking realism to a higher level, the integration of pedestrian simulations into our existing framework represents another compelling avenue. SUMO offers the capability to simulate pedestrian transportation, thereby introducing new variables into the traffic ecosystem. The introduction of pedestrian-specific traffic lights and pathways adds a layer of complexity that challenges traffic light control strategies.

This expansion necessitates the refinement and sophistication of reward functions and state representations. Balancing the needs of vehicles, public transport, and pedestrians becomes a multifaceted optimization problem. The ability of agents to process and respond to this wealth of information effectively becomes a fascinating research endeavor. One potential approach is the creation of distinct agents responsible for specific aspects of traffic control, with an overarching manager making decisions based on their outputs. This hierarchical structure offers a promising direction for handling intricate traffic scenarios.

9.3 Exploring Reward System Variations

Diving deeper into the realm of reinforcement learning, future work could entail the exploration of various reward systems and state representations. The flexibility of reward design allows for the experimentation with alternative combinations of state attributes. However, it is crucial to strike a balance, as overly complex reward structures may challenge the ability of agents to discern the underlying optimization objectives.

Additionally, the examination of pre-existing reward functions designed for similar agents merits consideration. The evaluation of these established reward systems within our framework can provide insights into their adaptability and effectiveness in optimizing traffic light control.

9.4 Exploring Action Space Variations

As previously discussed in Section 3.2.2, the RLight framework offers two distinct variations of the action space: cyclic and acyclic. In the context of our experiments, we initially adopted the cyclic action space, which involves predefined phase durations and allows the agent to transition from one phase to another within these predetermined time intervals.

However, our exploration doesn't stop there. It is intriguing to delve into the acyclic action space, an alternative approach where the agent possesses the liberty to select when to switch between phases without being constrained by predefined time intervals. This investigation into the acyclic action space can provide valuable insights into the flexibility and adaptability of the RLight framework in urban traffic signal control scenarios.

9.5 Contributions to the RESCO Repository

All of these future endeavors should ideally be conducted within the framework of the RESCO repository. By contributing to this public open-source repository, we can not only advance our own research but also benefit the broader scientific community. Improving the RESCO repository with enhanced simulation capabilities, additional traffic elements, and innovative control strategies ensures that the research conducted here serves as a valuable resource for other researchers, fostering collaboration and knowledge exchange.

In conclusion, the future of this research presents an array of exciting opportunities to push the boundaries of traffic flow optimization. These potential avenues offer the promise of more realistic simulations, advanced control strategies, and valuable contributions to the research community.

References

[1] M. Abadi and et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, 2016.

[2] L. N. Alegre. Sumo-rl, 2019.

[3] J. Ault, J. Hanna, and G. Sharon. Learning an interpretable traffic signal control policy. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2020)*. International Foundation for Autonomous Agents and Multiagent Systems, May 2020.

[4] James Ault and Guni Sharon. Reinforcement learning benchmarks for traffic signal control. In *Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021) Datasets and Benchmarks Track*, December 2021.

[5] Chacha Chen, Hongyu Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yilin Xiong, Kewei Xu, and Zongzhang Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3414–3421, 2020.

[6] T. Chu, J. Wang, L. Codecà, and Z. Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1086–1095, 2019.

[7] L. Codeca and J. Härri. Monaco sumo traffic (most) scenario: A 3d mobility scenario for cooperative its. In *SUMO 2018, SUMO User Conference, Simulating Autonomous and Intermodal Transport Systems, May 14-16, 2018, Berlin, Germany*, 2018.

[8] Seung Bae Cools, Carlos Gershenson, and Bart D'Hooghe. Self-organizing traffic lights: A realistic simulation. In *Advanced Information and Knowledge Processing*, pages 41–50, 2008.

[9] K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31:591–656, 2008.

[10] Yasuhiro Fujita, Prabhat Nagarajan, Toshiki Kataoka, and Takahiro Ishikawa. Chainerrl: A deep reinforcement learning library. *Journal of Machine Learning Research*, 22(77):1–14, 2021.

[11] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

[12] Sierk Kanis, Daan Bloembergen, Laurens Samson, and Tim Bakker. Deep reinforcement learning in traffic signal control. *arXiv preprint arXiv:2109.07180*, 2021.

[13] D. M. Levinson. Speed and delay on signalized arterials. *Journal of Transportation Engineering*, 124(3):258–263, 1998.

[14] S. C. Lobo, S. Neumeier, E. M. Fernandez, and C. Facchi. Intas-the ingolstadt traffic scenario for sumo, 2020.

[15] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[16] Jian Ma and Fan Wu. Feudal multi-agent deep reinforcement learning for traffic signal control. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 816–824, 2020.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[18] S. S. Mousavi, M. Schukat, and E. Howley. Traffic light control using deep policy-gradient and value-function-based reinforcement learning. *IET Intelligent Transport Systems*, 11(7):417–423, 2017.

[19] A. Paszke and et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[20] T. T. Pham, T. Brys, M. E. Taylor, T. Brys, M. M. Drugan, P. Bosman, M.-D. Cock, C. Lazar, L. Demarchi, and D. Steenhoff. Learning coordinated traffic light control. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-13)*, volume 10, pages 1196–1201, 2013.

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[22] S. M. A. Shabestary and Baher Abdulhai. Deep learning vs. discrete reinforcement learning for adaptive traffic signal control. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 286–293, 2018.

[23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[24] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (2nd ed.)*. MIT Press, 2018.

[25] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[26] A. Tirachini. Estimation of travel time and the benefits of upgrading the fare payment technology in urban bus services. *Transportation Research Part C: Emerging Technologies*, 30:239–256, 2013.

[27] Thomas Urbanik, Alison Tanaka, Bailey Lozner, Eric Lindstrom, Kevin Lee, Shaun Quayle, Scott Beaird, Shing Tsoi, Paul Ryus, Doug Gettman, et al. *Signal Timing Manual*. Transportation Research Board, 2015.

[28] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117*, 2019.

[29] Marco A. Wiering. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pages 1151–1158, 2000.

[30] H. Zhang, S. Feng, C. Liu, Y. Ding, Y. Zhu, Z. Zhou, W. Zhang, Y. Yu, H. Jin, and Z. Li. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The World Wide Web Conference*, pages 3620–3624, 2019.

[31] Dongbin Zhao, Yujie Dai, and Zhen Zhang. Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):485–494, 2011.

[32] G. Zheng, Y. Xiong, X. Zang, J. Feng, H. Wei, H. Zhang, Y. Li, K. Xu, and Z. Li. Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1963–1972, 2019.

[33] Guanjie Zheng, Xinshi Zang, Nan Xu, Hua Wei, Zhengyao Yu, Vikash Gayah, Kai Xu, and Zhenhui Li. Diagnosing reinforcement learning for traffic signal control, 2019.