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

Contents

1. Introduction

1.1 Overview

Traffic congestion is a persistent global issue, impeding daily commutes as a result of the ever-increasing urban population and transportation demands in cities worldwide [10][20]. One major contributor to this problem is the delay caused by red lights at intersections, where traffic signals typically operate on fixed-time schedules regardless of actual traffic conditions [15]. While such systems are effective in heavily congested areas, they often prove inefficient for low traffic density scenarios, resulting in unnecessary delays and fuel wastage [15].

Recent technological advancements have introduced the Adaptive Traffic Signal Control System, which utilizes sensors embedded in roads to synchronize traffic signals, thus responding to real-time traffic conditions [10]. However, this system's feasibility and cost-effectiveness have been questioned due to the need for embedded road infrastructure and power sources [10]. Additionally, optimizing traffic signal control to minimize delays while ensuring system stability remains a challenge [15].

This thesis aims to address these challenges by proposing a Traffic Control System based on reinforcement learning (RL), an artificial intelligence framework that learns optimal decision policies through continuous adaptation to real-time traffic scenarios. By moving away from fixed-time schedules and incorporating RL, we seek to develop an intelligent traffic control system that efficiently manages traffic flow, reduces environmental impact, such as air pollution and fuel wastage, and enhances road safety [15]. The research focuses on a 4-way intersection, analyzing incoming traffic density to optimize traffic signal control and improve overall transportation efficiency 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 primary objectives of this thesis encompass a comprehensive investigation into optimizing urban traffic flow through a multi-faceted approach. These objectives are designed to address the complexities of traffic management, improve realism in simulations, and assess the performance of cutting-edge algorithms and baseline controllers. The key objectives are as follows:

1.3.1 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.

1.3.2 Generation of Realistic Traffic Patterns

To enhance the realism of the simulations, real-world traffic patterns are essential. This objective involves collecting real-time traffic data from the chosen location and meticulously recording the timing and behavior of traffic lights. The gathered data will then be integrated into the simulation environment to replicate actual traffic conditions.

1.3.3 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.

1.3.4 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.

1.3.5 Comparative Analysis and Conclusions

Upon completing the simulations and experiments, the results from the various traffic signal control algorithms and baseline controllers will be rigorously analyzed and compared. The objective is to draw meaningful conclusions regarding the effectiveness of each approach in optimizing urban traffic flow. The research aims to provide insights into the potential for intelligent traffic management systems to alleviate congestion, improve efficiency, and reduce environmental impacts.

By achieving these objectives, this thesis seeks to contribute valuable knowledge to the field of urban traffic optimization and provide practical recommendations for enhancing traffic signal control systems in real-world urban settings.

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., $\pi : S \rightarrow A$. 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.

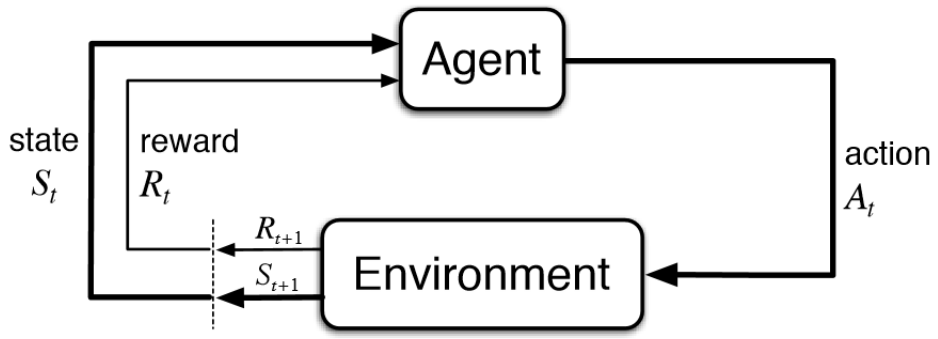


Figure 1: Reinforcement Learning Framework

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 [14].

- **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 [18].

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.

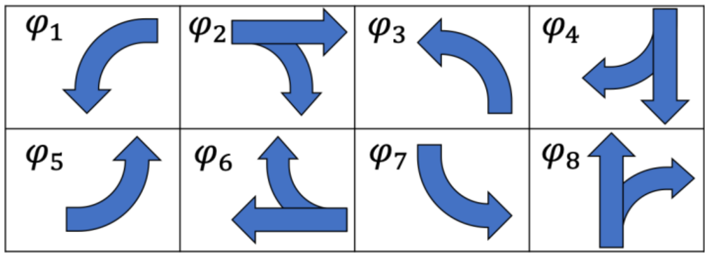


Figure 2: Example of Phases at a Signalized Intersection[4]

At each discrete time step, a signal controller is tasked with selecting a combination of non-conflicting phases to enable. The objective is to optimize a long-term objective function, which may vary depending on specific goals and constraints. In the context of Reinforcement Learning (RL)-based controllers, the signalized intersection environment is commonly modeled as a Markov Decision Process (MDP), with 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 [13] or solely the waiting time of the first vehicle in the queue [19].
- **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 [13], 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 [21], (minus) total delays imposed by the intersection [19], (minus) waiting time at the intersection [13], 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.[22]. 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 [17][8][13].

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.

State and Action Space

RESCO accommodates a wide range of sensing assumptions, including advanced sensing capabilities [7]. Users can select subsets of state features based on specific sensing assumptions. Features include information such as stopped vehicles' queue length, the number of approaching vehicles, total waiting time for stopped vehicles, and more, at the level of state, intersection, and lane. Additionally, users can define the effective sensing distance during initialization.

The action space in RESCO encompasses sets of non-conflicting phase combinations, following the methodology described in Section 2.2 of the RESCO documentation [7]. By default, actions are chosen for the next 10 seconds of simulation, with the first 3 seconds reserved for yellow signals, if necessary.

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.

Benchmark Control Tasks

The signal control benchmark tasks in RESCO are based on two well-established SUMO scenarios: "TAPAS Cologne" and "InTAS" [17, 11]. 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).

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.[13]
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[9], and pressure sensing[23].
   - (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, 13, 23].

In the case of IDQN, IPPO, and MPLight, the implementation of the learning algorithm is invoked directly from the ChainerRL [9] and the Preferred RL [9] libraries that is successor of ChainerRL, and customized to align with my specific map and requirements.

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. [14], 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. [18], 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

MPLight[5] is a traffic light control system that utilizes the concept of pressure to coordinate multiple intersections efficiently. It operates by considering the pressure, which is the difference in queue lengths from incoming lanes of an intersection and the queue length on a downstream intersection's receiving lane. MPLight is designed to optimize traffic flow and reduce congestion in urban environments.

In MPLight, pressure serves as a critical metric for traffic signal coordination. It is calculated as the difference between the queue lengths of vehicles waiting to enter an intersection and the queue length on the downstream intersection's receiving lane. By considering pressure, MPLight aims to balance the traffic load across multiple intersections.

Chen et al. introduced MPLight as an approach to traffic light control that leverages reinforcement learning techniques. They utilized Deep Q-Networks (DQN) as the underlying framework for making traffic signal decisions. In this setup, a DQN agent is shared across all intersections.

In MPLight, pressure is not only used as a coordination metric but also as both the state and reward for the DQN agent. The state of the agent at a given time step includes information about the pressure values for all relevant intersections. The reward signal is derived from pressure differences and is used to guide the learning process of the DQN agent.

Chen et al.[5] reported significant improvements in traffic flow and travel times when implementing MPLight compared to existing methods. Specifically, MPLight achieved up to a 19.2% improvement in travel times over the next best compared method, 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)

FMA2C introduces managing agents, which operate at a higher level of hierarchy compared to workers. Each managing agent is responsible for a specific region or area within the traffic network. These managing agents oversee multiple workers and have the responsibility of optimizing traffic flow within their assigned regions.

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

Managing agents are trained to optimize traffic flow within their assigned regions. They receive high-level traffic-related goals and objectives, such as minimizing congestion or maximizing traffic throughput. The managing agents use these goals to make region-level decisions.

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

FMA2C employs various coordination mechanisms between managing agents and workers to ensure effective traffic signal control. These mechanisms may include communication of high-level goals, reward sharing, and coordination through a central mechanism.

4.4.4 Performance Improvement

FMA2C aims to improve traffic flow and reduce congestion by introducing a hierarchical framework that allows for coordinated decision-making at both the region and intersection levels. By aligning the actions of workers with the goals of managing agents, FMA2C seeks to optimize traffic signal timings efficiently.

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)[12] 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)[12] 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)[12], 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[22], 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 introduction of CityFlow not only holds promise for optimizing traffic signal control but also opens avenues for various large-scale transportation research studies, such as vehicle routing through mobile apps and traffic jam prevention. Furthermore, CityFlow may serve as a benchmark reinforcement learning environment for transportation studies, similar to OpenAI Gym. The authors also express their intention to enhance the simulator by calibrating simulation parameters using real-world observations, thereby generating both fast and ”real” data samples.

This paper explores the capabilities and potential applications of CityFlow, emphasizing its contribution to addressing urban traffic control challenges and advancing 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)[22] 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.

For my thesis on optimizing urban traffic flow using reinforcement learning-based traffic light control, it was crucial to select a simulation tool that not only provides a robust and well-validated framework but also allows for the accurate representation of urban traffic scenarios. SUMO's extensive calibration, evaluation, and support for various real-world scenarios make it the ideal choice for my research objectives.

5.2 NetEdit

NetEdit is a powerful network editing tool developed as a part of the SUMO (Simulation of Urban Mobility)[12] 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.

NetEdit is designed to facilitate the creation and modification of road networks for traffic simulation purposes. It offers a user-friendly graphical interface that allows researchers, urban planners, and traffic engineers to define, edit, and refine road networks with ease. This tool is an essential component in the SUMO ecosystem, enabling users to customize network layouts, road geometries, traffic light configurations, and 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

NetEdit played a pivotal role in the creation of the "Vake" map network and subsequent traffic simulations. The "Vake" map is a significant case study in urban traffic optimization. Researchers leveraged NetEdit's capabilities to design a detailed and representative road network for the Vake district, incorporating real-world data and 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 [16].

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

In the pursuit of optimizing urban traffic flow through reinforcement learning-based traffic light control, my thesis leverages the capabilities of two prominent deep learning frameworks: PyTorch and TensorFlow. These frameworks play pivotal roles in the development and training of various intelligent agents within my research.

PyTorch, renowned for its dynamic computational graph and extensive support for neural network development, serves as the foundation for most of our intelligent agents. Specifically, the following agents are implemented using PyTorch:

- **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.

TensorFlow's static graph compilation and support for distributed computing are valuable for training these agents, particularly in large-scale and performance-critical scenarios.

The combination of PyTorch and TensorFlow allows us to harness the strengths of both frameworks to address various aspects of urban traffic flow optimization through reinforcement learning. These frameworks, together with my custom-developed agents, form the core of my research methodology.

5.4 Hardware Setup

The success of any computational experiment, especially those involving complex algorithms and simulations in the field of Artificial Intelligence, relies heavily on the underlying hardware infrastructure. In this section, we provide a detailed overview of the hardware setup utilized for my research, emphasizing its capabilities and constraints.

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 important to mention that the hardware used, specifically my personal laptop equipped with the aforementioned CPU and GPU, reflects a budgetary constraint. While these components have proven capable, advanced research in AI often requires access to high-performance computing clusters or specialized hardware designed for deep learning tasks. Despite these limitations, our dedication and commitment have enabled us to make significant progress in optimizing urban traffic flow using reinforcement learning-based techniques.

In conclusion, our hardware setup, comprising the Intel Core i5-9300H CPU and the NVIDIA GeForce GTX 1660 Ti Mobile GPU, has served as the workhorse for our research, albeit with inherent limitations. The continuous experimentation over a three-week period demonstrates our commitment to advancing the field of urban traffic flow optimization, even within the constraints of personal hardware resources.

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.

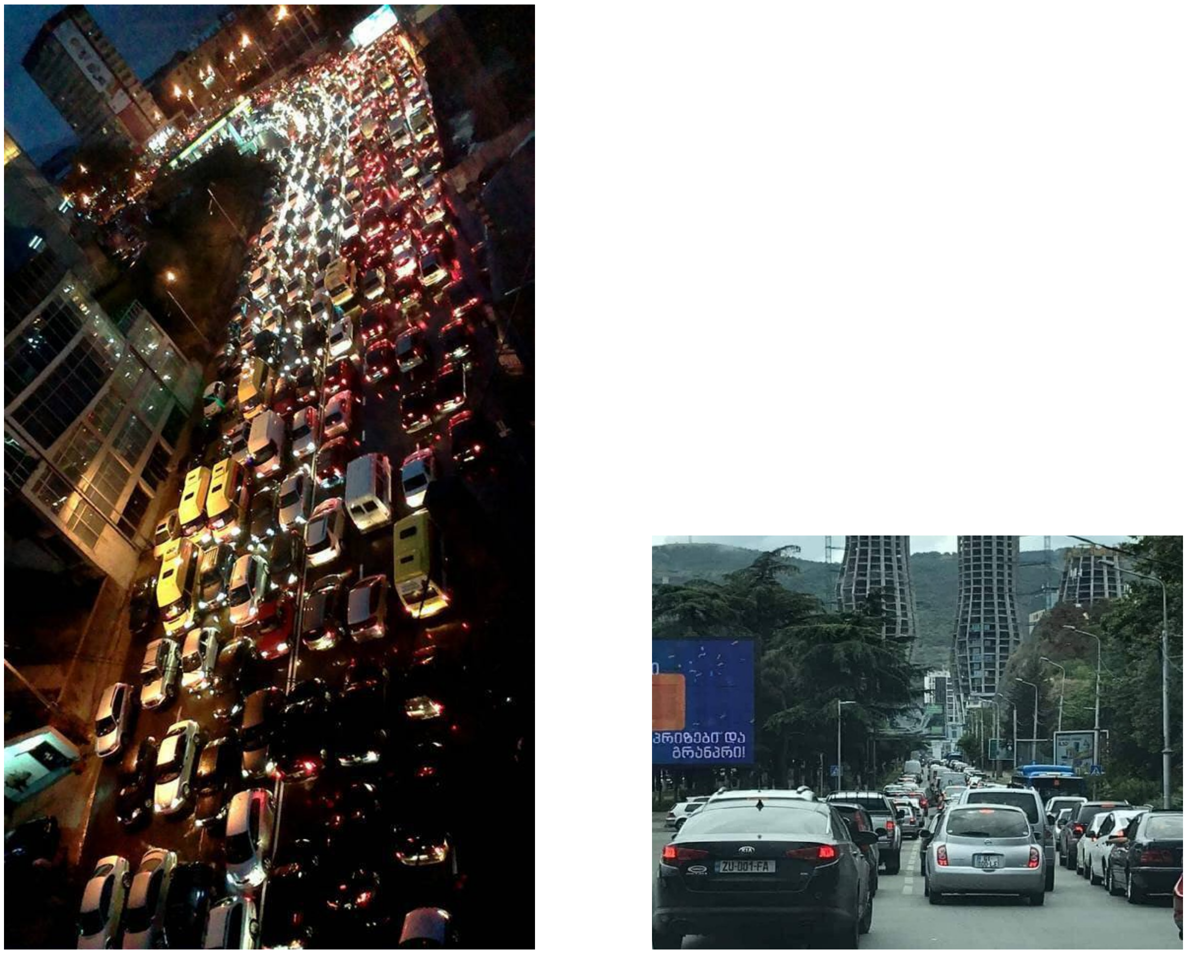


Figure 3: Typical Traffic Scene on Vake Street

Figure 3 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.

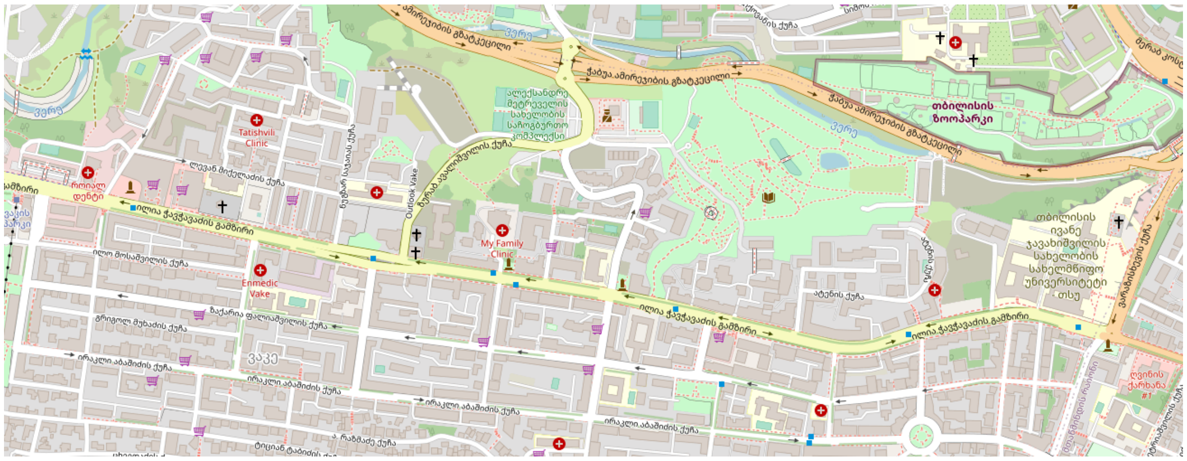


Figure 4: Vake Map from OpenStreetMap

As depicted in Figure 4, 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.

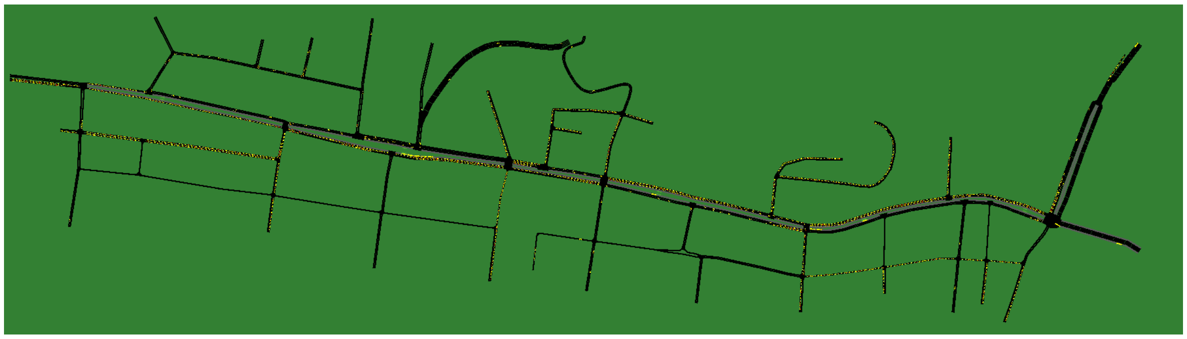


Figure 5: Vake Map for SUMO Simulation

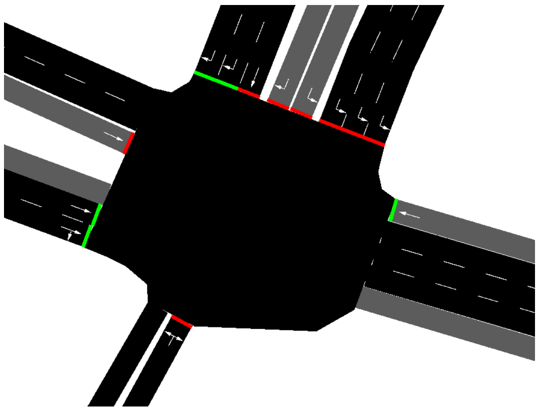
As seen in Figure 5, the result of our painstaking efforts is a map that faithfully replicates the shape and characteristics of the Vake area. This refined map serves as the groundwork for our subsequent traffic light control optimization experiments.

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

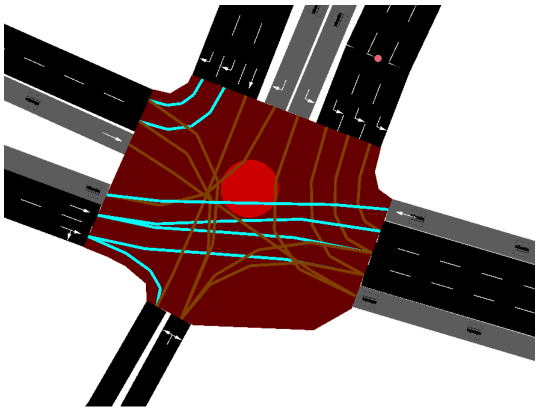

(b) Lane Directions

Figure 6: Traffic Light 1 (TL-1)

In Figure 6, 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 6 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.

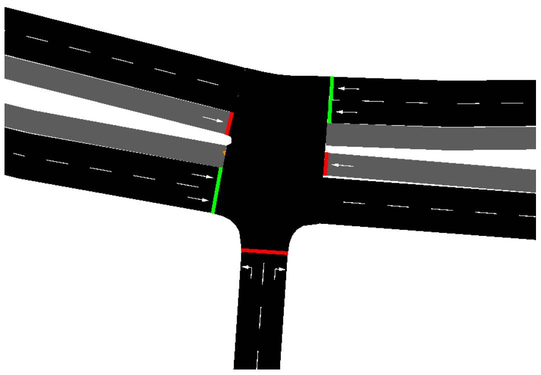
Bus Lane Configuration

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

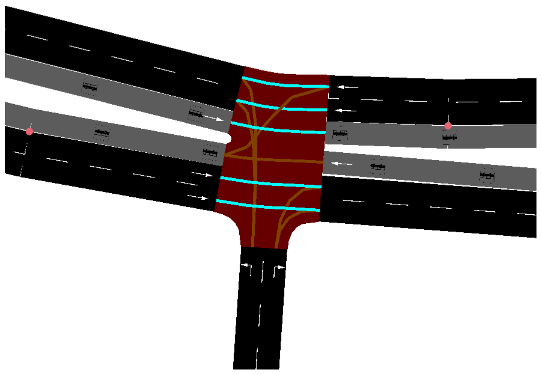
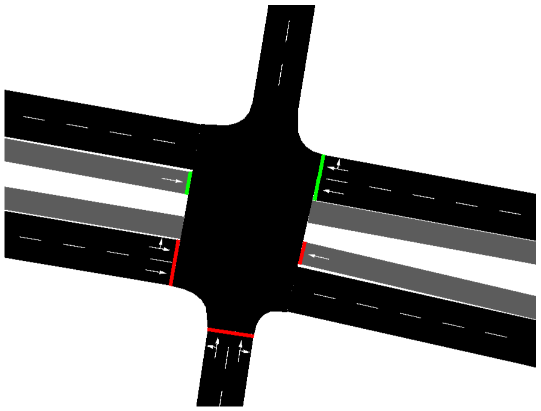

(b) Lane Directions

Figure 7: Traffic Light 2 (TL-2)

Comparing Traffic Light 2 in Figure 7 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

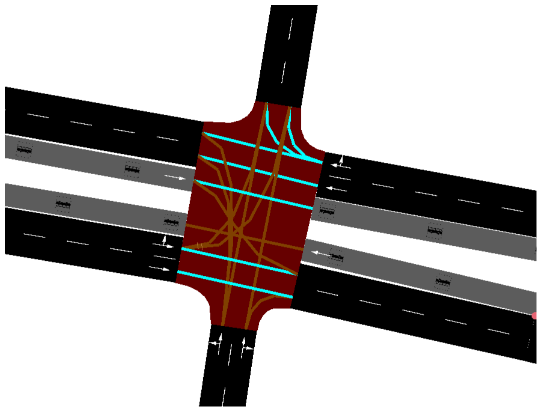
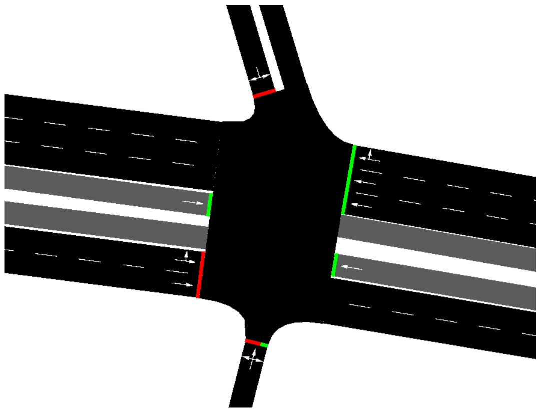

(b) Lane Directions

Figure 8: 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

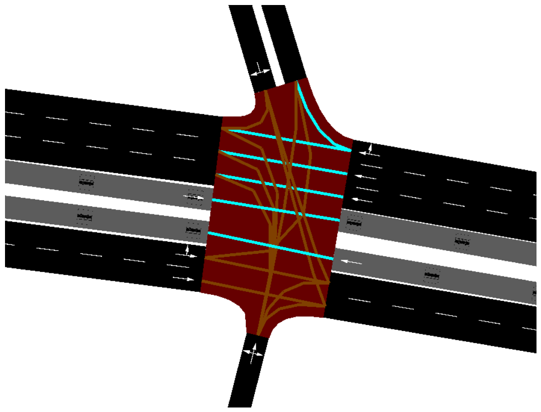

(b) Lane Directions

Figure 9: 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:

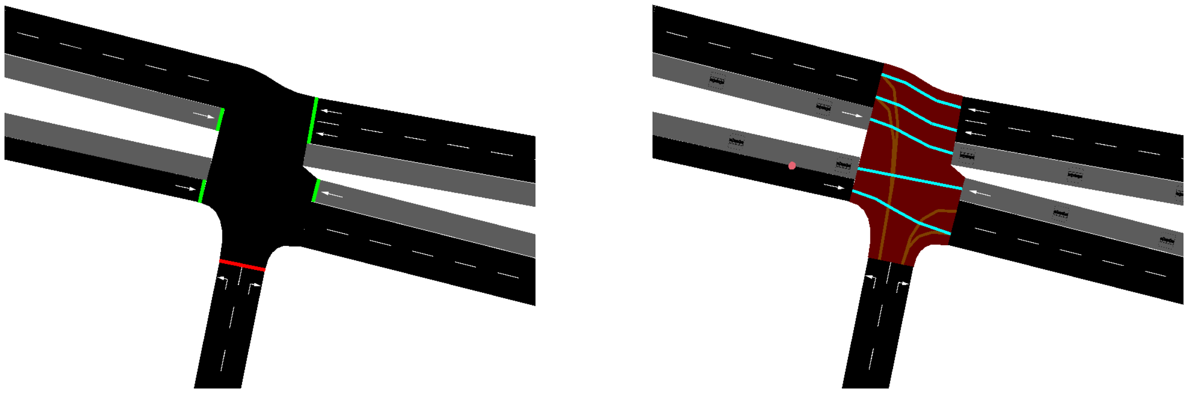

(a) Intersection Overview (b) Lane Directions

Figure 10: 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:

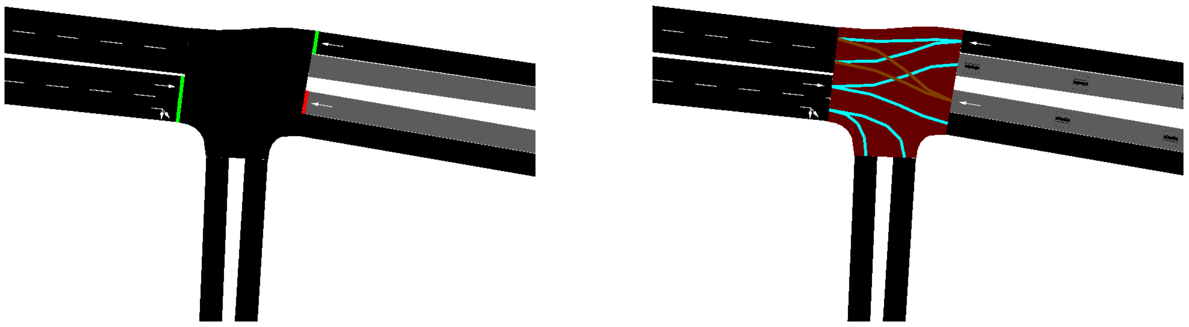

(a) Intersection Overview (b) Lane Directions

Figure 11: 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.

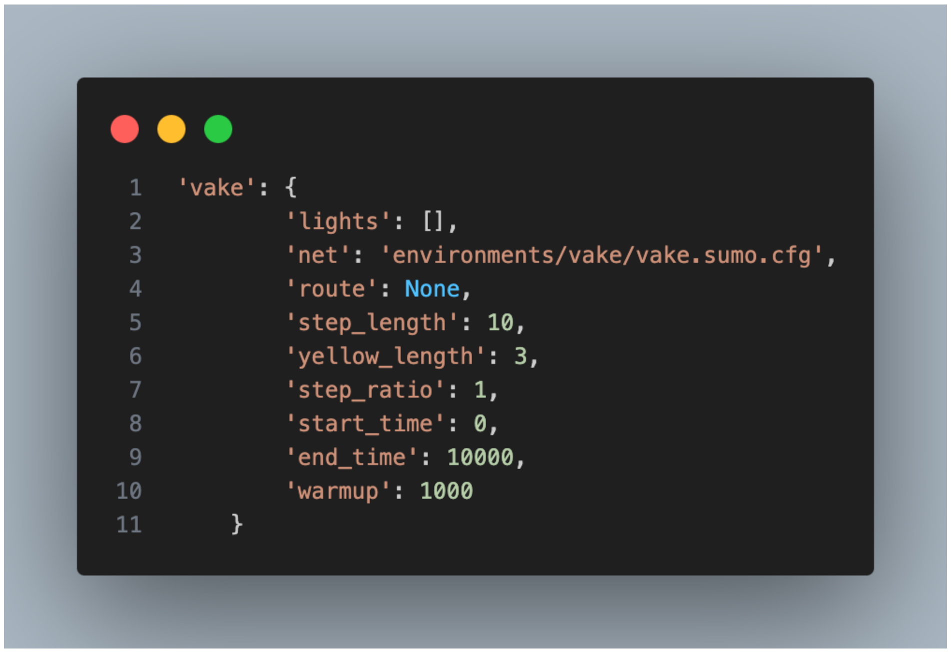
6.5.1 Map Config

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 12:

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': {
        'lights': [],
        'net': 'environments/vake/vake.sumo.cfg',
        'route': None,
        'step_length': 10,
        'yellow_length': 3,
        'step_ratio': 1,
        'start_time': 0,
        'end_time': 10000,
        'warmup': 1000
    }
```

Figure 12: Map Config of Vake

The Signal Config, as depicted in Figure 13, 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 14. 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.

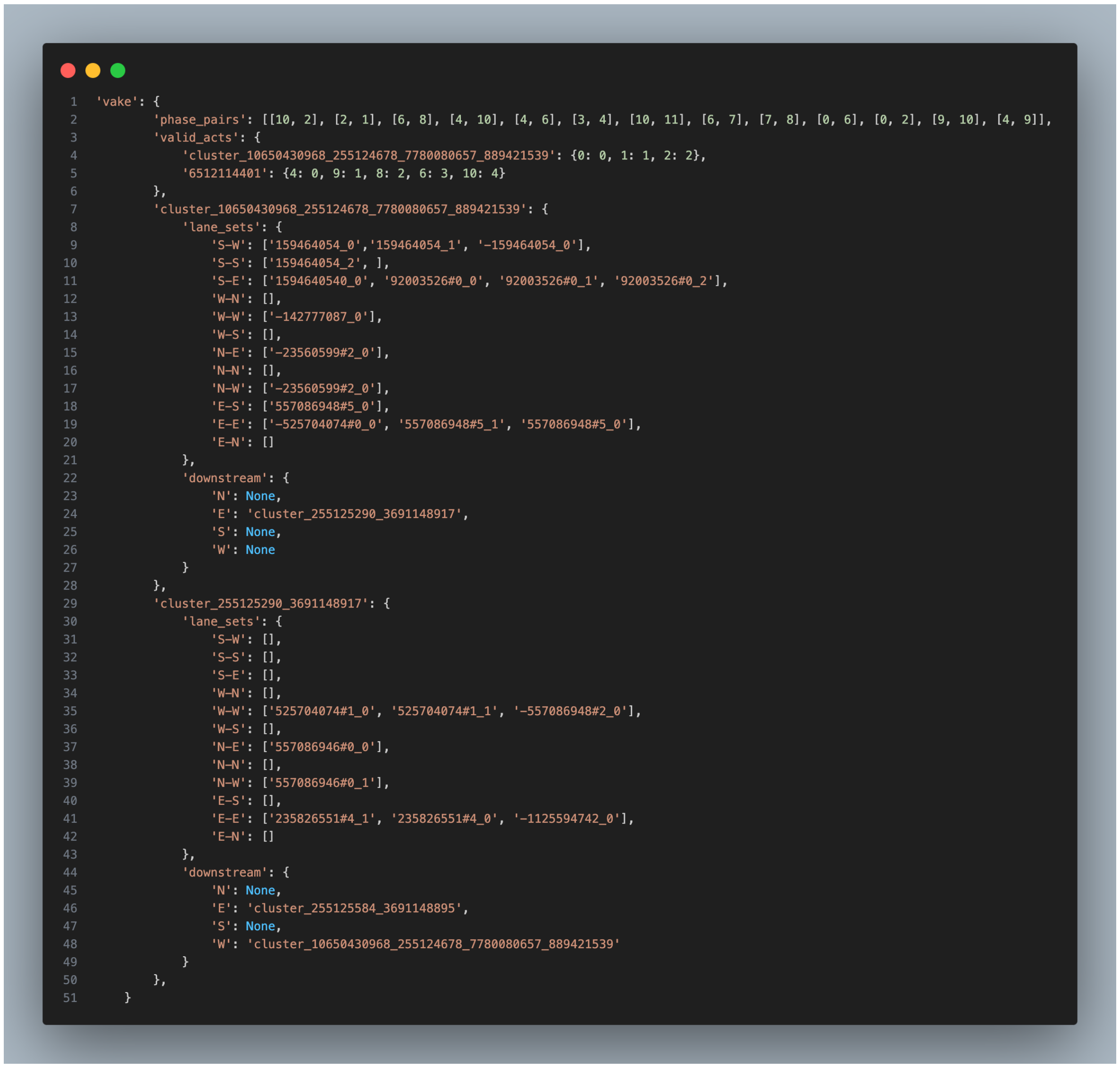
The configuration of the Signal Config posed significant challenges, mainly due to the lack of comprehensive documentation and explanations regarding its setup. Despite opening issues in the RESCO repository, obtaining guidance proved elusive, necessitating thorough investigation and experimentation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

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.



```python
'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'
            }
        },
    }
```

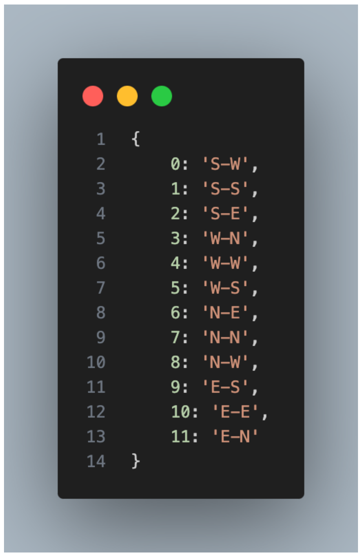
Figure 13: Signal Config of Vake

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.



```
{
    0: 'S-W',
    1: 'S-S',
    2: 'S-E',
    3: 'W-N',
    4: 'W-W',
    5: 'W-S',
    6: 'N-E',
    7: 'N-N',
    8: 'N-W',
    9: 'E-S',
    10: 'E-E',
    11: 'E-N'
}
```

Figure 14: Indexes of Traffic Flows

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.

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 provide insights into the rewards and states representations used for each controller. Additionally, 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 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.

7.1.1 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.

7.1.2 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.

7.2 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.

7.2.1 Results

Empty

7.3 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.

7.3.1 Results

Empty

7.4 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.

7.4.1 Results

Empty

7.5 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.

7.5.1 Results

Empty

7.6 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.

7.6.1 Results

Empty

7.7 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.

7.7.1 Results

Empty

7.8 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.

7.8.1 Results

Empty

7.9 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.

7.9.1 Results

Empty

7.10 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.

7.10.1 Results

Empty

7.11 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.

7.11.1 Results

Empty

7.12 Comparison

Empty

8. Conclusion

9. Future Work

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

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.