STAR: **S**mart **T**raffic **A**daptive **R**egulation

Applied Optimization Report

(Sambit Sahoo, Aditya Sinha, Adheesh Trivedi)

April 28, 2025

ABSTRACT

Urban traffic congestion is a big issue in India specifically, often using inefficient fixed-time traffic signal systems that fail to dynamic and uneven traffic patterns. This project proposes a data-driven, adaptive traffic signal control framework aimed at minimizing vehicle wait times and improving overall intersection efficiency. We explore three adaptive strategies: traditional tabular Q-Learning (QL), Deep Q-Learning (DQL) leveraging neural networks for scalable state-action approximation, and the decentralized Self-Organizing Traffic/ Lights (SOTL) approach. We apply a optimization formulation devleoped by us to SOTL, in order to minimize the waiting times.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to **Dr. P.B. Sujit** for giving us the opportunity to work on this project and for his invaluable guidance throughout the semester. His mentorship greatly helped us in understanding the core aspects of the problem and in developing effective solutions.

We are thankful to all our teaching assistants **Rajeev Ranjan Dwibedi**, **Atul Kumar**, **Agniva Banerjee** and **Kumar Katyayan Jaiswal** for their consistent support and insightful feedback, which played a key role in refining our approach. We are also thankful to **Akshat Singh** for his inputs in the constraints part.

A special mention to **Aadarsh** and **Kirti Agarwal** for their unique creative ideas and contributions which significantly enhanced our performance and presentation of the project.

Contents

1 Introduction

1.1 Motivation

Growing up in a bustling urban city, I never questioned the efficiency of traffic signals. With lanes equally busy throughout the day, fixed-time signals worked seamlessly, ensuring smooth traffic flow. Every cycle felt justified, balancing the needs of all directions without unnecessary delays. The system, though rigid, rarely felt inconvenient because congestion was evenly distributed.

That perspective changed when I visited a small city with uneven traffic distribution. Some lanes were nearly empty, while others had long queues, yet the traffic lights followed the same fixed schedule. I found myself stuck at a red light for over 240 seconds, waiting for an empty road to "clear." It was frustrating and inefficient—clearly, the system wasn't designed for such imbalances. That moment sparked my curiosity: how could traffic signals be optimized to adapt dynamically, minimizing unnecessary wait times while keeping intersections efficient?

1.2 Problem statement

Urban Traffic has become a persistent and growing challenge, leading to increased travel times, elevated fuel consumption, and heightened carbon emissions. At the heart of this issue lies inefficient traffic signal control, particularly at intersections, where traditional fixed-time signal plans fail to account for dynamic and fluctuating traffic conditions.

This project aims to develop an intelligent traffic signal control system that optimizes signal timings dynamically using a data-driven and learning-based framework. The proposed system leverages both real-time traffic data and historical traffic patterns to anticipate vehicle flows and make informed decisions on signal transitions.

- A baseline approach that uses constant signal durations irrespective of traffic conditions. This traditional method serves as a reference for evaluating the performance of adaptive models.
- **Q-Learning (QL):** A tabular reinforcement learning strategy where the agent learns an optimal signal-switching policy through trial-and-error interaction with the traffic environment. The Q-table is updated based on the observed states (vehicle counts on each lane) and received rewards (e.g., negative queue lengths), allowing the system to gradually adapt to changing conditions.
- **Deep Q-Learning (DQL):** An extension of QL that uses a deep neural network to approximate the Q-values. This method allows for generalization in larger state spaces and can handle more complex traffic scenarios. By learning from both real-time data and past experiences, the DQL model enhances decision-making in dynamic environments where the state-action space is too large for tabular methods.
- **Self Organinizing Traffic Lights (SOTL):** The Self-Organizing Traffic Light (SOTL) method, introduced by Gershenson (2005), is a decentralized, adaptive traf-

fic control strategy that relies on local vehicle detection to manage signal changes. Unlike traditional fixed-time or centrally coordinated systems, SOTL allows each intersection to make independent decisions based on the number of approaching vehicles at red lights. Each red light maintains a counter that accumulates the number of cars approaching over time. When this counter exceeds a predefined threshold, the signal switches from red to green, enabling responsive phase changes based on traffic demand.

2 Literature Study

The efficient management of urban traffic has long been a key area of research in intelligent transportation systems (ITS). Traffic congestion at intersections is one of the primary contributors to delays, increased fuel consumption, and emissions. To address these challenges, several traffic signal control strategies have been developed and studied over the years, ranging from traditional fixed-time systems to modern AI-driven adaptive approaches.

Traffic signal control was initially set at fixed time intervals, but this fixed-time control cannot effectively adapt to changing traffic flows. Therefore, the Adaptive Traffic Light Control (ATLC) technique has been developed to alleviate traffic congestion through the dynamic modification of signal timing. The Sydney Coordinated Adaptive Traffic System (SCATS) and Split Cycle Offset Optimizing Technique (SCOOT) are effectively implemented in many contemporary cities worldwide [7][8]. Investment is required for the deployment of sensors or manual modifications to enable the intelligent systems to collect data. The advent of artificial intelligence (AI) has enabled the collection and processing of a multitude of traffic flow data. Additionally, researchers have devised numerous novel techniques for traffic signal control, with extensive experimentation demonstrating that deep reinforcement learning (DRL) is a highly effective approach[3].

Traffic signal control methods based on deep reinforcement learning have received widespread attention. Zeng et al.[11] combined a recurrent neural network (RNN) with a DQN (i.e., DRQN) and compared its performance with that of a traditional DQN in partially observed traffic situations. Xie et al.[10] proposed the Information Exchange Deep Q-Network (IEDQN) method, which eliminated the communication problem between agents. Tunc et al.[9] proposed a novel approach to traffic signal timing at intersections, utilizing a combination of deep Q-learning algorithms and Fuzzy Logic Control (FLC), assisted by fuzzy logic. The approach focuses on objective evaluation metrics such as traffic congestion, air pollution, and waiting time to assess simulation results.

Numerous researchers have also started addressing practical issues related to traffic signal control, such as traffic flow, unexpected events, environmental pollution, and potential system failures, among others. Babatunde et al.[1] proposed an adaptive traffic signal controller method based on fuel-game theory that applies Nash Bargaining (NB) in an n-player cooperative game to determine the optimal phase assignment that takes into account future traffic signal control needs. The results demonstrate that this method is effective in reducing CO emissions and fuel consumption. Ounoughi et al.[5] proposed a traffic signal control method that combines future noise prediction with the deep duel-

ing Q-network algorithm, EcoLight Plus, to reduce CO2 emissions, noise levels, and fuel consumption.

There are several Adaptive Traffic light controllers without Q-Learning involved. Self-Organizing Traffic Light (SOTL)[2] control method is used as a baseline and integrate our ILP formulation of waiting time minimization to it and check the results on 4 lane junction network with increasing traffic demands.

For this comparative study, we inspired from the work of Ouyang et al. [6]

3 Controllers

In this section, we begin by outlining the fundamental concepts behind the Q-learning and Deep Q-Network (DQN) algorithms, along with their respective signal control frameworks. Following this, we present the design of the experimental environment. We then define the state space, action space, and reward structure used in our implementation. Lastly, We explain the "SOTL-request" control for traffic light with pseudocode of the controller.

3.1 Reinforcement Learning

3.1.1 Q-Learning

Q-learning is a reinforcement learning algorithm that enables a single agent to learn control policies and make decisions to optimize system performance by interacting with the environment through trial and error. The agent learns the action-value function (Q-function), which has three main advantages:

- it models the system's performance as a whole rather than focusing on individual influencing factors,
- it does not require prior knowledge of the environment's dynamics, and
- it does not require the transition probability matrix.

At each time step Q-learning is a reinforcement learning algorithm that allows an agent to learn optimal control policies through interaction with an environment, without requiring prior knowledge of the system's dynamics or transition probabilities.

At each time step t, the agent performs the following steps:

- **State Observation:** The agent observes the current state s_t, which belongs to the set of all possible states S.

$$s_t \in S$$

- **Action Selection:** Based on the observed state s_t, the agent selects an action a_t from the action space A.

$$a_t \in A$$

- **Reward Reception:** After performing the action a_t, the agent receives an immediate reward r_{t+1}, which depends on the current state s_t, the selected action a_t, and the resulting next state s_{t+1}.

$$r_{t+1} = R(s_t, a_t, s_{t+1})$$

- **State Transition:** The environment transitions to a new state s_{t+1} based on the current state and the selected action.

$$s_{t+1} = f(s_t, a_t)$$

- **Q-value Update:** The agent updates the Q-value for the state-action pair (s_t, a_t) using the following update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

  where α is the learning rate and γ is the discount factor.

Here, $\alpha \in [0, 1]$ represents the learning rate, which controls how much newly acquired information overrides old information. The term $\gamma \max_{a \in A} Q_t(s_{t+1}, a)$ denotes the discounted reward, where $\gamma \in [0, 1]$ reflects the agent's preference for future rewards over immediate rewards.

It is important to note that the size of the Q-table grows exponentially with the size of the state space $|S|$ and the action space $|A|$, especially in complex operating environments. This exponential growth leads to the curse of dimensionality and results in longer convergence times when learning the optimal policy.

3.1.2 Deep Q-Learning

The Q-learning algorithm learns the optimal policy by maximizing the expectation of future rewards, while DQN learns the Q-function by minimizing the loss function. The main idea of DQN is to improve the accuracy of Q-function by exploiting the representation capabilities of deep neural networks to provide a deeper understanding of states and actions. It learns high-level features of states and actions by using multiple hidden layers to be able to better predict future rewards. The DQN updates the parameters of the evaluated network by the following Equation (2). During training, the DQN updates the Q-function by constantly trying different actions and updating it based on reward feedback. It uses a backpropagation algorithm to optimize the parameters of the Q-function so that it can better predict future rewards [40]. Algorithm 1 represents the steps of a DQN algorithm. The loss function used in the Deep Q-Network (DQN) is defined as follows:

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

where $Q(s', a')$ is the action-value estimated by the target network, $Q(s, a)$ is the action-value estimated by the evaluation network, ω^- represents the parameters of the target network, and ω represents the parameters of the evaluation network. This loss function helps minimize the temporal difference (TD) error by adjusting the weights of the evaluation network through backpropagation.

Unlike the Q-learning control architecture, the DQN inputs real-time traffic state information (vehicle speed, number of waiting vehicles, delay time and traffic flow capacity) into the Q network and outputs the Q value corresponding to each signal decision action. According to the action selection policy, the signal decision action with the maximum Q value is selected and finally fed back to the environment. This process is repeated until the optimal cumulative reward is achieved. Figure 2 shows the architecture of signal control based on the DQN algorithm.

Listing 1: DQN Algorithm

```
Initialize replay memory D to capacity N
Initialize observation steps S and total steps T ← 0
Initialize action-value function Q with random weights θ
Set target network weights: θ⁻ ← θ

for each episode n = 1 to N
    Observe initial state s_0 of the traffic light
    Choose initial action a_0 ~ π_θ(s_0)
    for t = 1 to K
        Observe new state s_t and reward r_t
        Store transition (s_{t-1}, a_{t-1}, r_{t-1}, s_t) in D
        Update step counter: T ← T + 1
        if T > S
            // Sample random minibatch of transitions
            (s_{t-1}, a_{t-1}, r_{t-1}, s_t) from D

            //Set target value:
            Q(s_t, a_t) = r_{a,t+1} + γ max_{a'} Q(s_{a,t+1}, a'; θ⁻)

            Perform gradient descent to update weights θ
        end if
        if T mod C = 0
            Update target network: θ⁻ ← θ
        end if
    end for
end for
```

3.2 SOTL

The SOTL (Self-Organizing Traffic Light) control method operates on a decentralized decision-making framework, where each traffic signal manages its own timing based on the number and flow of vehicles approaching it. A key feature of the approach is the use of a counter at each red signal, which increments in proportion to the number of incoming

vehicles. Once this count exceeds a predefined threshold, the signal changes from red to green, allowing high-demand lanes to be serviced more responsively.

To maintain operational stability, SOTL includes a minimum green phase duration that prevents lights from switching too rapidly in high-density traffic. Moreover, to avoid disrupting vehicle groups—or platoons—already traversing an intersection, the system checks for the presence of vehicles on cross streets before changing the light. This mechanism ensures smooth and uninterrupted passage for moving platoons. In cases where a significant number of vehicles are queued at a red light, the system is permitted to override this constraint, thereby preventing prolonged delays on busier approaches.

Listing 2: SOTL-request control pseudocode

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

3.2.1 General Formulation

We formulate the problem as follows:
In this model we consider an intersection with N lanes, where each lane i has a maximum green light duration denoted by ϕ_i. The density of vehicles in each lane is represented by a_i, which is the number of vehicles arriving per second. During the time when the green light is off for a particular lane, vehicles start to accumulate. We define this non-green duration for the i^{th} lane as:

$$\psi_i = \sum_{j \neq i} \phi_j + N\phi_{\text{yellow}}$$

which includes the green durations of all other lanes and the yellow light duration repeated for each lane. The objective is to minimize the total vehicle accumulation in all lanes,

given by the function.This expression quantifies the cumulative accumulation of vehicles when the lights are red or yellow for each lane, and by minimizing it, we aim to optimize traffic flow and reduce congestion at the intersection.

Our goal is to optimize the durations of the traffic lights in such a way that vehicle congestion is minimized. The objective function is defined as:

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

where $\psi_i = \sum_{j \neq i} \phi_j + N\phi_{\text{yellow}}$ represents the time duration for which the i^{th} lane does not have a green signal, and a_i is the arrival rate (vehicles per second) for lane i.

Constraints

- $\phi_i \geq 5$: This ensures that the duration of the green light for any lane is not unrealistically short.
- Soft constraint: $\phi_i \geq 2a_i\psi_i$ for all $i = 1, 2, \ldots, N$. This constraint is based on the idea that we want each lane to have sufficient green time to clear the queued vehicles. Since it may not always be feasible to fully clear the queue, this condition is treated as a soft constraint.
- $\phi_i \in Z$: Each green light duration must be an integer value. This transforms the problem into an Integer Linear Programming (ILP) problem.

3.2.2 4 Lane Formulation

We use the previous general formulation with certain assumptions:

1. The junction has 4 lanes.
2. The vehicle densities of all the 4 lanes are equal i.e $a_1 = a_2 = a_3 = a_4 = \text{a}$.

This reduced our objective function to just minimize $\phi_i = \phi$ as everything else becomes constant. So, minimum value of ϕ that will satisfy the constraints is:

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

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

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

4 Results

In this section we will give the results for Reinforcement learning and SOTL approach. We used mean queue length and mean waiting time as benchmark for both the approaches. The simulations are build with SUMO[4].

4.1 Reinforcement Learning

4.1.1 Network Specification

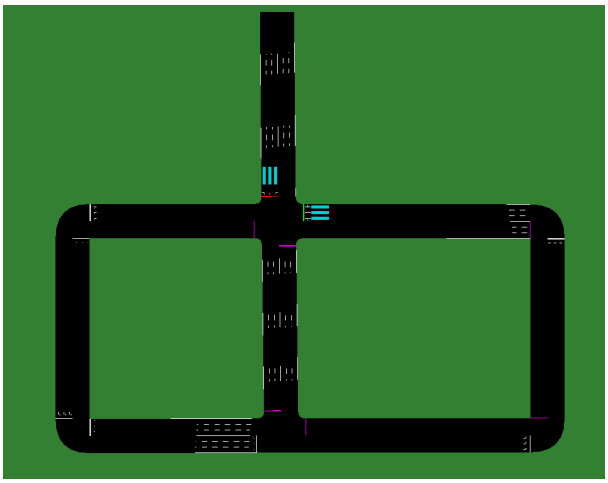

Figure 1: 4-Way Intersection Road Network used for simulation in SUMO

The simulation is conducted on a complex 4-way intersection road network (Figure 1), modeled in SUMO, with three lanes per direction. Each direction consists of a single incoming and outgoing lane, creating a realistic and challenging traffic environment. Traffic signals at the intersection are centrally controlled using either a fixed-time strategy, Q-learning (QL), or Deep Q-Network (DQN) based adaptive strategies. To mimic real-world traffic dynamics, vehicles are generated at random intervals, introducing variability in traffic flow and testing the adaptability of each control method.

4.1.2 Result comparison

Figure 2: Cumulative Reward Comparison across Fixed Time, Q-Learning, and Deep Q-Learning

The performance of different traffic control strategies is evaluated using cumulative reward as a metric, as shown in figure above. The Fixed Time approach, which does not adapt to real-time conditions, yields the lowest cumulative reward, indicating poor performance in minimizing delays and managing traffic flow. Q-Learning improves upon this by gradually learning a better switching policy through interaction with the environment,

but it remains limited in scalability and convergence. Deep Q-Learning (DQL), with its ability to generalize across larger state spaces using neural networks, consistently outperforms the others, achieving a higher and more stable cumulative reward—demonstrating its effectiveness in learning adaptive traffic signal policies.

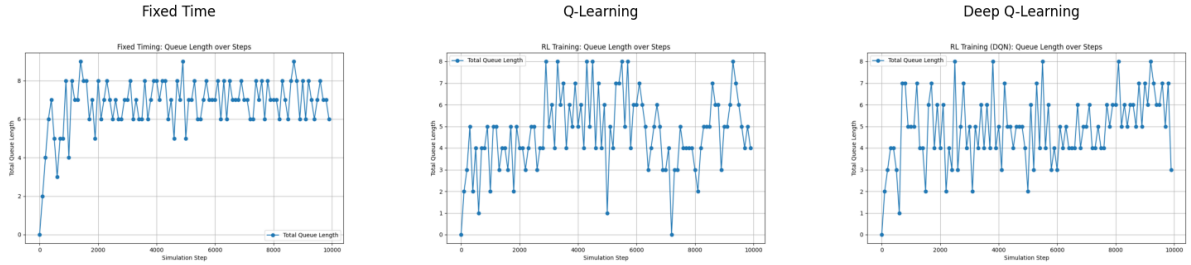

Figure 3: Queue Length Trends across Fixed Time, Q-Learning, and Deep Q-Learning

Above presents the average queue length trends for each method. Fixed Time control results in persistently high queue lengths due to its inflexible nature. In contrast, Q-Learning begins to show improvements by reducing queue sizes over time as it learns from traffic patterns. Deep Q-Learning offers the most significant benefit, maintaining consistently lower and more stable queue lengths throughout the simulation, highlighting its superior adaptability and effectiveness in managing real-time traffic congestion.

4.2 SOTL

4.2.1 Network Specification

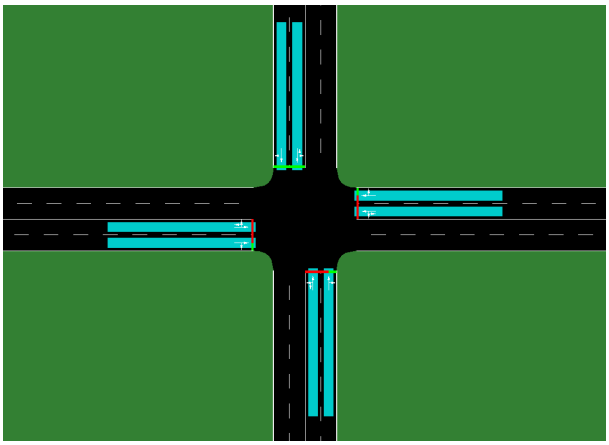

Figure 4: 4 Lane network in SUMO

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

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

Traffic lights for detecting traffic violation.

4.2.2 Fixed green times

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

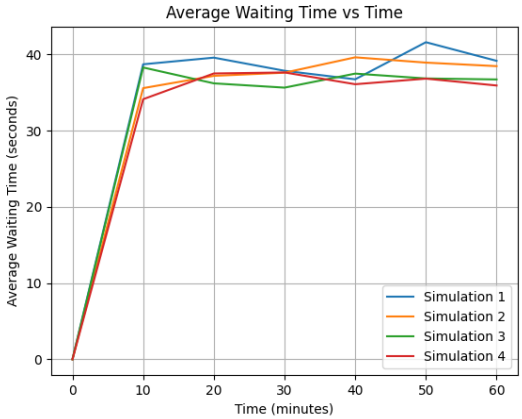

Figure 5: Simulation 1, 2, 3, 4 for 250, 350, 450, 550 vehicles per hour respectively

4.2.3 Optimized parameters

By substituting the values for vehicles per hour in the formula derived in Section 3.2.2, we get the following results:

a	ϕ
250	5
350	6
450	12
550	44

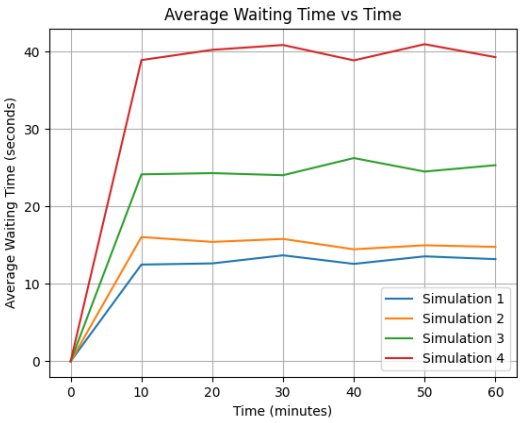

Figure 6: Simulation 1, 2, 3, 4 for 250, 350, 450, 550 vehicles per hour respectively, with optimized parameters

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

References

[1] Jibril Babatunde, Osama A Osman, Aleksandar Stevanovic, and Nemanja Dobrota. Fuel-based nash bargaining approach for adaptive signal control in an n-player cooperative game. *Transportation research record*, 2677(10):451–463, 2023.

[2] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe. Self-organizing traffic lights: A realistic simulation. *Advances in applied self-organizing systems*, pages 41–50, 2008.

[3] Mustafa Coşkun, Abdelkader Baggag, and Sanjay Chawla. Deep reinforcement learning for traffic light optimization. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 564–571. IEEE, 2018.

[4] Daniel Krajzewicz. Traffic simulation with sumo–simulation of urban mobility. *Fundamentals of traffic simulation*, pages 269–293, 2010.

[5] Chahinez Ounoughi, Ghofrane Touibi, and Sadok Ben Yahia. Ecolight: Eco-friendly traffic signal control driven by urban noise prediction. In *International Conference on Database and Expert Systems Applications*, pages 205–219. Springer, 2022.

[6] Chen Ouyang, Zhenfei Zhan, and Fengyao Lv. A comparative study of traffic signal control based on reinforcement learning algorithms. *World Electric Vehicle Journal*, 15(6):246, 2024.

[7] Arthur G Sims and Kenneth W Dobinson. The sydney coordinated adaptive traffic (scat) system philosophy and benefits. *IEEE Transactions on vehicular technology*, 29(2):130–137, 1980.

[8] Aleksandar Stevanovic and Peter T Martin. Split-cycle offset optimization technique and coordinated actuated traffic control evaluated through microsimulation. *Transportation Research Record*, 2080(1):48–56, 2008.

[9] Ilhan Tunc, Atakan Yasin Yesilyurt, and Mehmet Turan Soylemez. Different fuzzy logic control strategies for traffic signal timing control with state inputs. *IFAC-PapersOnLine*, 54(2):265–270, 2021.

[10] Donghan Xie, Zhi Wang, Chunlin Chen, and Daoyi Dong. Iedqn: Information exchange dqn with a centralized coordinator for traffic signal control. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[11] Peng Zeng, Hepeng Li, Haibo He, and Shuhui Li. Dynamic energy management of a microgrid using approximate dynamic programming and deep recurrent neural network learning. *IEEE Transactions on Smart Grid*, 10(4):4435–4445, 2018.