Robust Multi Agent Deep Q-Learning for Traffic Signal Control

Mathew K. John
UMass Amherst
Amherst, MA 01003
mkjohn@umass.edu

Ryan Fletcher
UMass Amherst
Amherst, MA 01003
rafletcher@umass.edu

Justin Fallo
UMass Amherst
Amherst, MA 01003
jfallo@umass.edu

Abstract

Current traffic control systems struggle to handle the intensity of traffic in urban areas. Decreasing road congestion by introducing novel methods for traffic control systems would reduce greenhouse gas emissions, the commute time of drivers, and the consumption of resources by vehicles. This paper explores an extension to an approach that uses a deep Q-network (DQN) with multi-agent reinforcement learning (MARL) for continuously adapting traffic signal control of a targeted region. A simulation of the Ingolstadt, Germany transportation network was used for experimentation. Although previous literature suggests assigning a DQN to each agent in the network gives the best results, this paper demonstrates that a single DQN that considers and controls all agents may perform better in realistic traffic scenarios.

1 Introduction

1.1 Background and purpose

The problem of road congestion causes longer commute times and higher transportation costs for drivers, and the emission of greenhouse gases such as carbon dioxide and nitrous oxide from vehicles is a leading cause of global warming. Currently, most traffic signal control systems use fixed time signal control (FTSC) plans because they are inexpensive and easy to implement. These plans use pre-set signal cycles determined by previous traffic data. Thus, they cannot best satisfy traffic demand at a particular time and location. This paper introduces an approach to control traffic signals based on live traffic patterns in a targeted region to combat road congestion.

1.2 Research scope and methodology

The intent of the project was to compare performance of three different approaches to traffic signal control: the FTSC approach, an approach from previous literature that assigned a DQN to each agent (multi-DQN, or MDQN), and the proposed approach that uses a single DQN for all agents (SDQN). (We believe SDQN to be more bio-plausible than MDQN because humans tend to notice and process general characteristics before focusing on finer details.) Steps were taken to enhance the realism of the simulation to improve results, but in doing so, our hardware became unable to support the increased computational requirements of the network. Thus, we were unable to train the latter two networks to convergence and instead trained each for three episodes – an episode is an epoch that may have a different traffic pattern than previous episodes. Three measures of performance were selected: system total waiting time (how much time the vehicles currently on the roads have spent at a standstill), system average vehicle speed, and total CO_2 emissions. The FTSC, MDQN, and SDQN schemes were compared to each other over three different traffic schedules to determine performance and robustness to changing traffic patterns.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

2 Literature review

2.1 Fixed time signal control

Fixed time signal control is a method used by traffic signal systems to regulate traffic. The timing of signal phases are predetermined such that the duration of each phase and the sequence of phases do not change in response to actual traffic conditions. Traffic engineers determine the appropriate phase intervals based on traffic volume, road geometry, and pedestrian activity. These time intervals are then programmed into the signal controller. Since the traffic signal cycles are a predetermined sequence of phases, the duration of each phase remains constant even if there are no vehicles or pedestrians present. Advantages of fixed time signal plans are it is simple and predictable. Plans do not require continuous monitoring or adjustment, and drivers can become familiar with signal patterns to know what to expect at each intersection.

2.2 Deep Q-Network

Reinforcement learning (RL) is a machine learning technique that involves three components: actions, rewards, and observations. Actions are what an agent can do, the reward reflects the success of an agent's action, and the observations represents the state of the environment. RL is used to solve sequential decision-making problems by learning how to behave in a given situation to maximize the reward. One of the predominant RL algorithms is a DQN. Q-learning is a learning algorithm that has a Q function that estimates a Q value for each state–action pair where the Q value determines whether to perform a specific action in a specific state. A DQN uses a deep neural network to estimate the Q function and compute the Q value so that it can solve problems with large state and action spaces. A DQN uses experience replay and target network techniques. Experience replay is a method in which an agent stores a sample of their experience with the environment in memory and randomly extracts it for learning. By removing high correlation of the training data, DQNs suppress overfitting and enable stable learning. Target network techniques use the target network to eliminate learning instability when using one network.

2.3 Prior traffic signal control studies using reinforcement learning

DQN-based traffic signal control models Park et al. developed two traffic signal control models using a DQN: an isolated intersection traffic signal control model and coordinated intersection traffic signal control model. The performance was evaluated by comparing the developed models to an optimal fixed-time signal model. A simulation environment was constructed using Vissim and the COM interface. The environment was so simple that applying the developed traffic signal control model to a real site could prove difficult. Thus, the model was limited by a lack of practical experience. The traffic signal control method for the models provides a signal timing plan based on traffic demand for the next cycle. At the end of a cycle, the traffic signal control model selects the optimal signal timing plan for the next cycle. Traffic demand for a cycle was described by the maximum queue length which is the maximum number of waiting vehicles in the cycle. For the RL algorithm, the state was described by the maximum queue length, the action was described by the signal timing plan selected for the next cycle, and the reward was described by the average stop delay. Thus, the models learned by selecting actions that minimized the average stop delay per cycle. Evaluation and validation showed that performance was superior to that of the optimal fixed-time signal plan in terms of average delay, average travel speed, and average number of stops for both models. However, they were trained on very small areas with a primitive simulation.

RL benchmarks for traffic signal control Ault and Sharon propose a toolkit for developing and comparing RL based traffic signal controllers that includes implementation of RL algorithms for signal control and benchmark control problems based on realistic traffic scenarios. The toolkit allows for comparison of RL based signal controllers while providing benchmarks for future comparisons. The paper compares the relative performance of current RL algorithms on the Cologne and Ingolstadt road networks. These road networks were chosen because they are well-accepted by the transportation community and include a congested downtown zone with multiple signalized intersections. The paper suggests that a deep Q-learning approach is best for more realistic signal layouts.

3 Development of a traffic signal control model

3.1 Selection and implementation of reinforcement learning algorithms

A DQN was implemented due to its success in previous literature. We employed experience replay and target network techniques to improve DQN performance. A double-ended queue of 4-tuples with maximum capacity 10,000 was used to implement the former. Each 4-tuple contained an action, reward, state, and subsequent state to represent an RL step. This implementation let us conserve memory, safely store new step reports, and remove old 4-tuples. At each step, 128 samples were randomly selected for learning. Although the data generated during traffic signal control simulation is large and storage is limited, we were able to avoid any problems with this implementation.

Multi agent reinforcement learning was implemented so agents could cooperate by sharing information about the impedance of vehicles at their traffic signal. This let agents make decisions within the context of the entire transportation network. Two MARL algorithms were developed. The first used one DQN for all agents (SDQN), and the second assigned a DQN to each agent with the agents cooperating with each other (MDQN). The structures of these algorithms are shown in Figure 1. The rewards received by each agent indicated the maximum traffic lane impedance weighted by the number of emergency stops made by vehicles in the controlled incoming lanes.

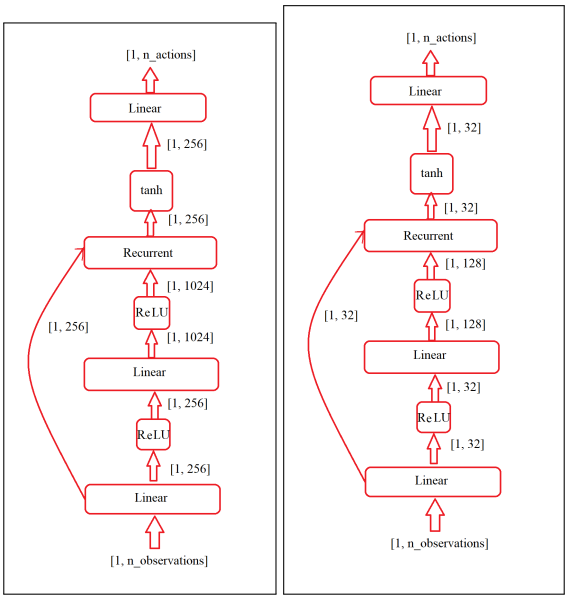

Figure 1: SDQN (Left) and MDQN (Right)

3.2 Development of simulation environment framework

We used Simulation of Traffic Mobility (SUMO) to simulate a traffic environment. According to TraCI's pydoc generated documentation, SUMO is able to calculate microscopic variables of vehicles such as speed, position, acceleration, various gas emissions, noise emissions and fuel consumption at each second or millisecond. SUMO also has various car following, junction, and lane change models. This variety enhances the realism of the simulation because it portrays driving behavior more accurately. We used the PyTorch, SUMO-RL, and TraCI packages to develop an RL-based TSC model. The PettingZoo package was used to implement MARL to control all traffic lights in the transportation network. The Ingolstadt transportation network was chosen because it is well-accepted by the transportation community and includes a congested downtown zone with multiple signalized intersections. RL agents observed the network's microscopic traffic flow by interfacing with TraCI in Python.

3.3 Development of a coordinated intersection traffic signal control model using a DQN

Three different traffic schedules in the virtual Ingolstadt transportation network were the test bed for TSC experiments. The network contained 12,138 activated junctions and 27,754 activated edges, 17 roundabouts, and 1,484 traffic lights. A rendering of the Ingolstadt network in SUMO is shown in Figure 2. (Due to our hardware/memory limitations, there were many thousands of additional junctions and edges that we had to leave inactive in order for SUMO to run properly.)

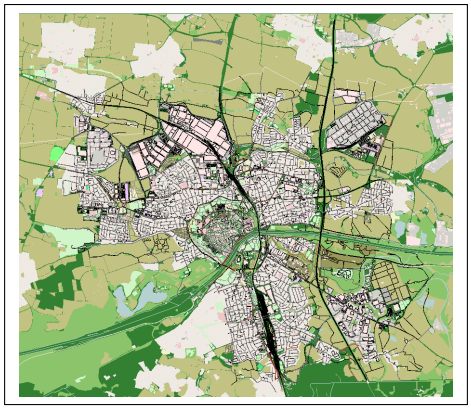

Figure 2: Virtual Ingolstadt transportation network in SUMO

4 Evaluation and validation of the developed traffic signal control model

4.1 Traffic signal control model evaluation and validation method

We selected three measures of effectiveness to evaluate model performance: system total waiting time, system average vehicle speed, and total CO_2 emissions. The measures were calculated at each step. The first two measures were selected for their use in previous literature. System total waiting time is the total time vehicles spent waiting in the transportation network in seconds, and average vehicle speed is the average speed of all vehicles in the transportation network in meters per second. The third measure was selected because CO_2 is the primary greenhouse gas emitted by human activities. Total CO_2 emissions is the mass of CO_2 emitted by all vehicles in the transportation network in kilograms. We compared the SDQN and MDQN schemes to the FTSC model to evaluate their performance. We also compared the MDQN and SDQN on each measure over three different traffic schedules to determine the robustness of the modified architecture.

4.2 Evaluation and validation of coordinated intersection traffic signal control model

Estimation, optimization and emulation methodology To evaluate the performance of the DQN based TSC models, we required a virtual environment created based on real-world traffic data and geographic data. The real-world traffic conditions and traffic light programs of the German city of Ingolstadt were emulated using estimations of the traffic conditions in the city. According to Harth and others, conducting traffic signal control experiments in the real world is expensive and consumes great amounts of time. Furthermore, real world traffic signal control experiments can be dangerous and their results may not be reproducible. In comparison, traffic signal control simulation is cheap, quick, safe and the results of experiments can be replicated. In addition to improving the validity of a traffic simulation, using real-world traffic conditions and real world geographic maps allows us to compare the results of real-world traffic signal control with simulated results. The traffic data scenario we used in our research paper was created in the SAVe:, SAVeNoW and KIVI research projects. OpenStreetMap was used to model a virtual transportation network based on the city of Ingolstadt. Real word traffic data from the city of Ingolstadt was transferred to SUMO. Although Ingolstadt uses actuated traffic signal control and prioritizes public transportation, the average green time of the traffic signals was used to create hourly fixed time traffic signal control programs for 75 of the traffic lights. The static programs of the rest of the traffic lights were manually created to handle traffic demand. Traffic flow through the virtual network was calibrated using regional traffic

statistics in Ingolstadt. Factors like the number of inhabitants in Ingolstadt and their preferences were considered when generating the traffic flow. Langer and others write that a genetic algorithm was employed to optimize 6 parameters of the car-following model and the junction model of the SUMO environment. The Krauss car-following model was determined to be the optimal car-following model in SUMO for the Ingolstadt transportation network.

Traffic scenarios simulated We simulated a 4 hour long traffic scenario from 6 AM to 10 AM, then 4am to 8am, then 8am to noon on Wednesday, September 16 2020 in Ingolstadt for each of the RL TSC models. The simulated network involved multi-modal transport like passenger vehicles, heavy-vehicles and bicycles. The simulation was slightly influenced by the COVID restrictions on September 16, 2020 in Ingolstadt. Although there are sidewalks, pedestrians won't be able to walk over crossings between several streets of the network. Therefore, the traffic scenario can't be used to simulate pedestrians crossing streets because the network is disconnected and we didn't simulate pedestrian movement.

Results The developed TSC models were evaluated by comparing the selected measures with the FTSC model and by comparing the selected measures among the developed TSC models. When comparing the simulation results of FTSC with the DQN based TSC models, it is evident that in early episodes the FTSC model handily (and expectedly) outperforms the DQN based TSC models in terms of total waiting time, mean speed and CO2 emissions. It is unknown how our models perform at or after convergence, because the more realistic traffic network and scenario require more computational power than we had access to. Previous literature indicates convergence after 50-60 epochs of 4 hours each for a single traffic schedule, so on three different traffic schedules we estimate convergence would occur by 180 episodes of 4 hours each, depending on implementation. This would have taken us roughly two to three months. However, again, we only have access to extremely subpar processing equipment. Using faster CPUs and any GPUs at all, this time may be greatly reduced. We were able to run the models' training at 50%-33% real-time speed. A reasonably expensive computer could certainly run the models' training at 100% real-time speed, using 24-hour episodes of actual live traffic data. (We believe 24 hour episodes would be optimal, because traffic tends to cycle over 24-hour periods.)

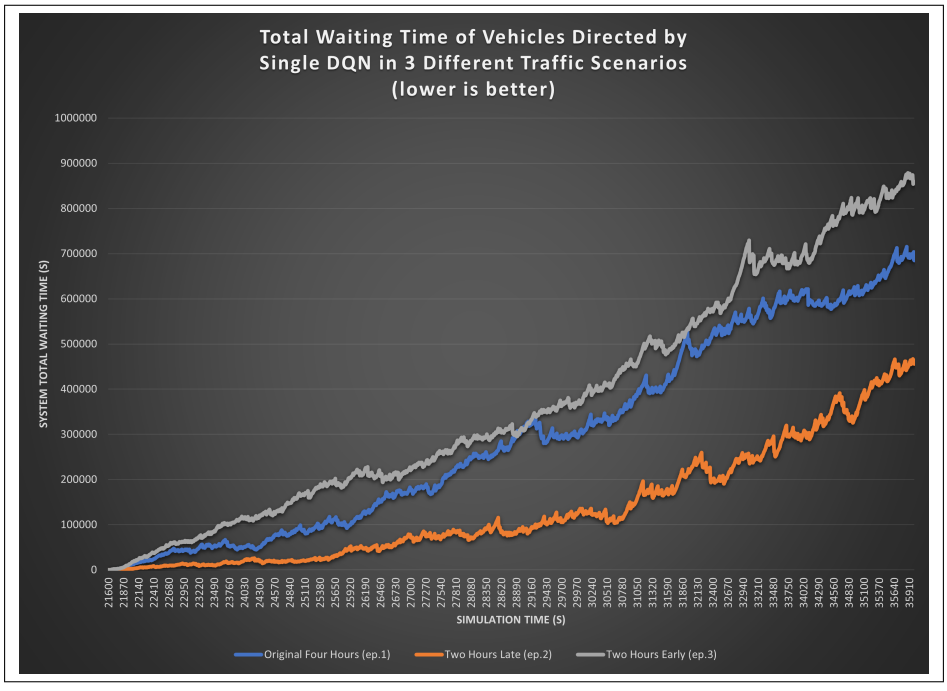

Figure 3: SDQN System total waiting time over 3 different traffic schedules

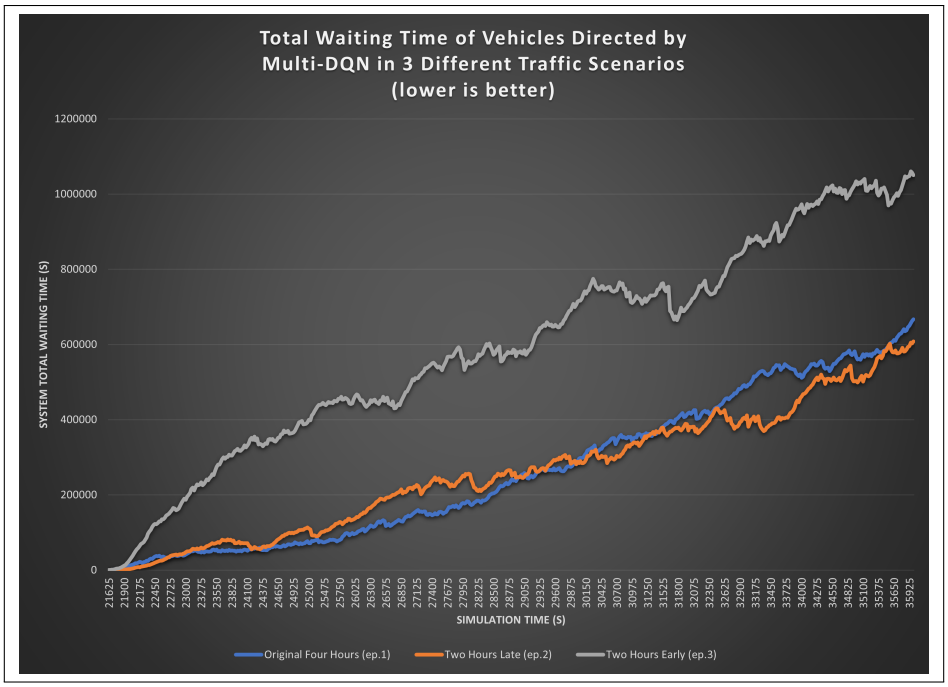

Figure 4: MDQN System total waiting time over 3 different traffic schedules

This is the most important result. Here, episode 1 (in blue) is run with the traffic data gathered in real life. Here the SDQN ends the episode with roughly 685,000 seconds of accumulated waiting time and the MDQN ends a little lower than that. There's the expected slightly better performance from the MDQN. Episode 2 (in orange) is that same traffic schedule, but with all vehicles shifted so that they enter the network two hours later. Here the SQDN ends the episode with roughly 450,000 seconds of accumulated waiting time, but the MDQN ends with roughly 600,000 seconds. Episode 3 (in grey) is the original traffic schedule shifted two hours earlier. Here the SDQN ends with roughly 860,000 seconds, while the MDQN ends with roughly 105,000,000 seconds.

Performance between episodes 1 and 2 should improve (total waiting time should drop) because the traffic is lighter. The SDQN shows this pattern. The MDQN, however, has nearly identical overall performance between episodes 1 and 2 and shows only a slight improvement towards the end. This indicates that it is simply re-learning the traffic patterns from scratch, meaning it's less robust in the face of the altered traffic schedule. This pattern repeats between episodes 2 and 3, with the SDQN showing better performance.

Table 1: System Total Waiting Time of RL TSC Models

System Total Waiting Time (s)			
Model	Median	IQR (Spread)	End Value
MDQN Episode 1	236257.5	351573.75	666890
SDQN Episode 1	282247.5	431005	684870
MDQN Episode 2	254630	279183.75	608225
SDQN Episode 2	216259.4708	198963.75	456110
MDQN Episode 3	585145	363851.25	1049960
SDQN Episode 3	307767.5	407943.75	859050

Table 2: Average Vehicle Speed of RL TSC Models

System Average Vehicle Speed (m/s)			
Model	Median	IQR (Spread)	End Value
MDQN Episode 1	5.63598075	1.056066959	4.96387364508767
SDQN Episode 1	5.664018245	0.985396938	5.24766053944322
MDQN Episode 2	5.595345481	0.637473808	5.62376648370313
SDQN Episode 2	5.796776957	0.512661368	5.85417229452004
MDQN Episode 3	4.863242178	0.350363935	4.77248450260839
SDQN Episode 3	5.077047416	0.373579431	4.88358465838296

Table 3: CO_2 Emissions of RL TSC Models

CO_2 Emissions (kg)			
Model	Median	IQR (Spread)	End Value
MDQN Episode 1	52.88434657	10.95811956	46780572.7648133
SDQN Episode 1	52.56692221	11.00044491	47727550.9873173
MDQN Episode 2	33.96151034	42.79278984	67567795.9644966
SDQN Episode 2	33.00780081	41.88415698	66225286.0905897
MDQN Episode 3	51.3320337	5.19934694	52893311.0303432
SDQN Episode 3	49.61298674	4.940244754	50741855.8790754

Although the median system total waiting time of the MDQN was better than the median system total waiting time of the SDQN by 45,990s in episode 1, the SDQN outperformed the MDQN in terms of median system total waiting time in the other 2 episodes by at least 315,748.0292s when combining together the reduction in median system total waiting time that the SDQN allowed for episodes 2 and 3. The SDQN defeated the MDQN in terms of median system mean speed and median total CO2 emissions in all 3 episodes. We didn't observe any trends when comparing the interquartile ranges of the three MOEs for both the MDQN and the SDQN.

In a previous iteration of the experiment, we trained the SDQN for an extra episode and compared the results we obtained with the FTSC model and the results of the same TSC model during episode 1. The FTSC model still outperforms the SDQN TSC model in terms of the median system total waiting time, system mean speed and total CO2 emissions. The median system total waiting time in the SDQN based TSC model improved at an encouraging rate that indicates it may eventually converge to perform at least on par with the FTSC model. The median system mean speed also improved in the extra episode, but the total CO2 emissions became worse. There was also a decrease in the standard deviation of the median system total waiting time, median system mean speed and total CO2 emissions, indicating it becomes more stable over time.

5 Conclusions and future studies

This paper discussed a solution to traffic congestion in urban areas. A DQN-based RL algorithm was chosen because of its extensive use in previous studies. Experiments were conducted in a microscopic simulation environment that was created with SUMO and PettingZoo, and TraCI was used to interface the environment. The test bed for experiments was the Ingolstadt transportation network. Each RL experiment was carried out for three episodes, each with a different traffic schedule. Prior literature suggests that the MDQN scheme performs better than the SDQN scheme after convergence by 50-60 epochs. However, their experiments were conducted on an Ingolstadt network with far less detail and with constant traffic patterns. Our experiments indicate that the MDQN scheme severely falters in the face of changing traffic patterns to which the SDQN scheme was robust. Thus, the SDQN scheme may benefit from never being denied learning (aka "frozen") even if it eventually converges. Therefore, this model should be further investigated since changing traffic patterns occur all the time in real traffic scenarios. The FTSC scheme outperformed both the SDQN and MDQN schemes in terms of system total waiting time, system mean speed, and total CO_2 emissions, but after convergence this is likely to not be the case.

There are several simple improvements that could be made to this experiment that we were unable to make due to hardware limitations. Testing behavior in much later episodes, such as 50-60 or 150-180 depending on the exact traffic pattern changes, would allow for performance evaluations near convergence. The system could also be run in real-time rather than at the 50%-33% speed we were forced to run the system at. The realism of the hyperparameters could also be improved. For example, the ideal step time is ≤ 0.25 seconds but ours had to be 5 seconds. It could also prove useful to experiment with transportation networks from other cities or with different traffic patterns to determine if the SDQN scheme could be employed in any urban location.

References

[1] S. Park, E. Han, S. Park, H. Jeong, and I. Yun, "Deep Q-network-based traffic signal control models," PLOS ONE, vol. 16, no. 9, p. e0256405, Sep. 2021, doi: https://doi.org/10.1371/journal.pone.0256405.

[2] J. Ault and G. Sharon, "RL Benchmarks for Traffic Signal Control." Accessed: May 06, 2023. [Online]. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/f0935e4cd5920aa6c7c996a5ee53a70f-Paper-round1.pdf

[3] J. Lee, J. Chung, and K. Sohn, "RL for Joint Control of Traffic Signals in a Transportation Network," IEEE Transactions on Vehicular Technology, vol. 69, no. 2, pp. 1375–1387, Feb. 2020, doi: https://doi.org/10.1109/TVT.2019.2962514.

[4] S. Liu, G. Wu, and M. Barth, "A Complete State Transition-Based Traffic Signal Control Using Deep RL," IEEE Xplore, Apr. 01, 2022. https://ieeexplore.ieee.org/document/9794168/ (accessed May 06, 2023).

[5] M. Harth, M. Langer, and K. Bogenberger. "Automated Calibration of Traffic Demand and Traffic Lights in SUMO Using Real-World Observations". SUMO Conference Proceedings, vol. 2, June 2022, pp. 133-48, doi:10.52825/scp.v2i.120.

[6] M. Langer, M. Harth, L. Preitschaft, R. Kates, and K. Bogenberger, "Calibration and Assessment of Urban Microscopic Traffic Simulation as an Environment for Testing of Automated Driving," IEEE Xplore, Sep. 01, 2021. https://ieeexplore.ieee.org/document/9564743 (accessed May 06, 2023).

[7] German Aerospace Center. (n.d.). Python: package traci, Retrieved from Simulation of Urban MObility website: https://sumo.dlr.de/pydoc/traci.html.

[8] United States, Environmental Protection Agency. (2023, April 13). Overview of Greenhouse Gases | US EPA. Retrieved from the United States, Environmental Protection Agency website: https://www.epa.gov/ghgemissions/overview-greenhouse-gases.