# Multiagent Reinforcement Learning for Traffic Signal Control

**Kevyn Kelso**
University of Colorado-Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO 80919
kkelso@uccs.edu

## Abstract

Transportation is a complex problem facing developed societies. It is riddled with logistical challenges extending beyond pollution and traffic congestion. The idea of applying reinforcement learning to transportation problems is not new but has great potential for improvement and research in the field. Contrary to the fixed algorithms currently deployed in most areas (Chaudhuri et al. 2021), reinforcement learning can make decisions in stochastic and uncertain environments characteristic of traffic problems. Many common reinforcement learning approaches have been applied, showing promise over other approaches, and will be compared below. This project aims to not only reproduce existing methods but also apply new ideas to improve performance concerning traffic throughput metrics described below. An integration of various ideas from the literature, combined to study what can be learned from reinforcement learning experimentation with regards to traffic control is the primary purpose.

## Introduction

Traffic Signal Control (TSC), a classic control problem affecting all drivers, holds exciting potential for reinforcement learning to address obvious societal problems including air pollution, decreases in speed, delays, and opportunity costs (de Almeida, Bazzan, and Abdoos 2022). Based on travel research conducted in urban areas, 12-55% of total commute time is caused by signalized intersections (traffic lights) (Ault and Sharon 2021). Moreover, modern reinforcement learning (RL) approaches suggest a potential 73% reduction in imposed delays compared to traditional approaches currently deployed (Ault and Sharon 2021). Discussions on traffic signal control contain many ideas to increase traffic efficiency in both the fixed algorithm and RL space. However, many advanced solutions come with expensive caveats. For instance, installing advanced sensors, reworking road paths, or replacing traffic lights with roundabouts (Alegre, Bazzan, and da Silva 2020). This project aims to study reinforcement learning approaches to TSC that can be deployed to existing metropolitan areas with minimal cost and infrastructure changes. In this paper, we will discuss the TSC problem in terms of the Markov Decision Process (MDP) paradigm; and how it is formally described. Existing methods will then be explored including what is deployed traditionally in most urban environments, and what RL methods could also be used. Then, our unique solution will be proposed based on information collected comparing a wide variety of methods used previously on the TSC problem (Hafiz and Bhat 2020) (Ault and Sharon 2021) (Ghanadbashi et al. 2023). Results will be discussed regarding the current status of the project showcasing what has been done thus far. Finally, a discussion of evaluation metrics, challenges overcome, and the timeline of the project will be discussed.



## Problem Formulation

Multiagent approaches to TSC are Partially Observable Markov Decision Processes (POMDPs) with non-stationary dynamics, placing them in one of the most difficult classes of problems for RL to solve (de Almeida, Bazzan, and Abdoos 2022) (Alegre, Bazzan, and da Silva 2020) (Choi, Yeung, and Zhang 1999). Non-stationary problems do not come with convergence guarantees of traditional MDPs and the partial observability can cause the agent to miss nuance in the state necessary for optimal policies (Choi, Yeung, and Zhang 1999) (Lee, Ganapathi Subramanian, and Crowley 2022). State aliasing is common in POMDPs where the agent finds two states that are different to be the same, resulting in inappropriate actions. Additionally, centralized agent approaches suffer from Bellman's curse of dimensionality, require unfeasible infrastructure modifications, and have shown suboptimal performance in simulation (Alegre, Bazzan, and da Silva 2020) (Ault and Sharon 2021).

### Environment Simulator

Accepted and widely used, Simulated Urban MObility (SUMO) environment is used for experimentation(Ault and Sharon 2021). SUMO-RL [1], an OpenAI PettingZoo API-compatible environment will be used for multiagent simulations (Terry et al. 2021). It is a Python wrapper that controls SUMO via its Traffic Control Interface (TRACI) API. A 4 x 4 grid structure 1 of traffic lights will be used as it is also widely used by other papers (de Almeida, Bazzan, and Abdoos 2022) and is easily understood. This re-

[1] Alegre, L. N. 2019. SUMO-RL. https://github.com/LucasAlegre/sumo-rl.

quires 16 independent agents. In addition to single intersection, and 4 x 4 grid experiments, the downtown portion of Colorado Springs is also simulated, using 203 independent agents to control each traffic intersection. As a future work item, larger portions of the city can be simulated to determine their effect on existing results. To simulate the entire area under the control of the Colorado Springs Signal Timing department, 712 independent agents are required. This scale would require more computing hardware than what is currently available for this project.

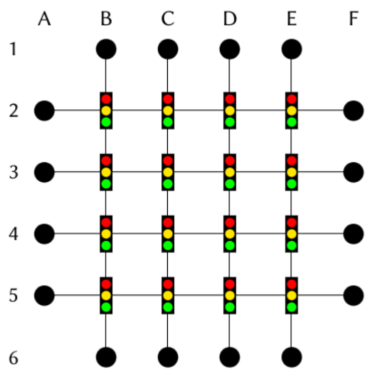


Figure 1: 4 x 4 intersection grid (Alegre, Bazzan, and da Silva 2020)

## State space

The state space for traffic signal control is a vector of various traffic-related parameters described below. In the TSC problem, the state space is typically modeled as equation 1 where each time step $t$ corresponds to five seconds of actual traffic dynamics, $\rho_1$ and $\rho_2$ are binary variables $\rho_1, \rho_2 \in \{0, 1\}$ indicating the state of the intersection lights, $g$ indicates if the light has been green for the minimum specified time, $L$ is the list of all lanes with density $\Delta_l$ which is the number of vehicles in each lane divided by the capacity of that lane. $q_l$ is the number of queued vehicles in each lane $l \in L$ (de Almeida, Bazzan, and Abdoos 2022).

$$s_t = [\rho_1, \rho_2, g, \Delta_1, \ldots, \Delta_L, q_1, \ldots, q_L] \tag{1}$$

In the simulation experiments mentioned below, equation 1 represents the state space for all agents. In reality, Colorado Springs has a variety of different state spaces depending on the intersection. Some intersections only have an inductive sensor for each road leading to the intersection and do not have per-lane granularity like equation 1. Some intersections in Colorado Springs implement Continuous Tracking Advance Detectors (CTAD) (Adv 2022) which can provide a similar state space to what is described in equation 1. The city is working to install these advanced detection sensors in more intersections as time goes on. Future work for this project will focus on making the state space match exactly what has been implemented in the field.

An approximation of the state space for downtown Colorado Springs shown in figure 2 has been obtained using OpenStreetMap (OSM) data [2]. OSM sources its data through a combination of publicly available government data and crowdsourcing. It is not guaranteed to be 100% accurate. In this project, it is used as a proof of concept, requiring more accurate information if agents are to be trained for field deployment.

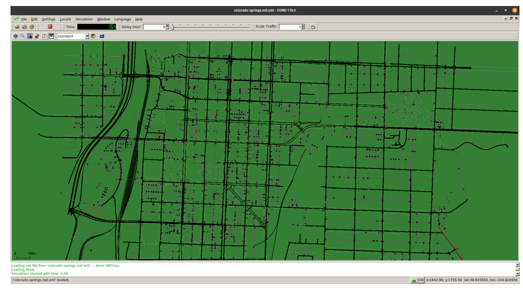

Figure 2: Downtown Colorado Springs Simulation

## Action space

The action space is best described in figure 3 where the green paths indicate where traffic can flow and the red paths indicate no traffic flow. The action space for the simplest type of intersection consists of four phases including a North-South, East-West, and two turning phases. These intersections are used in the 4x4 grid experiments. The intersections used for the Colorado Springs experiments vary between 2 and 12 possible phases. The phase information for each intersection in the Colorado Springs simulation is imported into SUMO from OSM data. To change the intersection into a different phase, a mandatory yellow phase $\phi$ precedes the phase change. In all the experiments below $\phi$ = 2 seconds. However, the legal value of $\phi$ is dictated by Manual on Uniform Traffic Control Devices (Federal Highway Administration 2023) and is based on many factors including the neighborhood, speed limit, and traffic visibility.

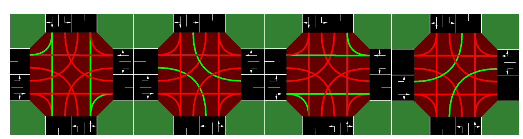

Figure 3: Traffic Light Phases [3]

## Reward function

In most papers, the cumulative delay reward function 2 is used where $D_t$ is the sum of all vehicles wait time 3. Other papers extend this idea to the sum of all vehicle wait times in the system to encourage cooperation between agents (Lee, Ganapathi Subramanian, and Crowley 2022). Furthermore, other sophisticated reward functions such as traffic pressure, augmented rewards based on intersection efficiency, and rewards based on ontology adherence have been explored (Ault and Sharon 2021) (Ghanadbashi et al. 2023).

[2]https://www.openstreetmap.org/

In this project, the cumulative delay reward function 2 is used. The cumulative delay used in the experiments below is the sum of all vehicles in the system instead of at the intersection level, where $V_t$ is the set of all vehicles present in the simulation. The idea behind this is to encourage the agents to work together rather than develop adversarial relationships. The mean cumulative delay for all vehicles in the system was also tried, yielding acceptable results.

$$r_t = D_t - D_{t+1} \tag{2}$$

$$D_t = \sum_{v \in V_t} d_t^v \tag{3}$$

## Existing Methods

Most popular RL methods have been applied to TSC including Independent Deep Q-Learning Networks (IDQN), Independent Proximal Policy Optimization (IPPO), MPLight (variant of DQN) (Chen et al. 2020), Feudal Multiagent Advantage Actor-Critic (FMA2C, MA2C) (Chu et al. 2019), State Action Reward State Action Lambda $SARSA(\lambda)$, TD methods (Reza et al. 2023), Self Adaptive Systems (SAS), and ontology-based RL models (Ghanadbashi et al. 2023). Based on the results from each of these papers, IDQN appeared to have the most promising performance which will be discussed further below. Fixed time, max pressure, and greedy algorithms are not RL algorithms but are kept here as a baseline to reference (Ault and Sharon 2021) (Chen et al. 2020). Our method is represented in bold font, showing good performance in the 4 x 4 grid scenario.

| Method | Average Cumulative Delay (seconds) |
|---|---|
| Fixed Time | 90.00[2] |
| Max Pressure | 70.00[2] |
| Greedy | 60.00[2] |
| IDQN | 30.74 |
| **Our Method** | 20.00 |
| IPPO | 36.15 |
| MPLight | 54.58 |
| MA2C | 38.07[1] |
| FMA2C | 42.26 |
| $SARSA(\lambda)$ | 18.00[2] |
| Ontology | 17.00[2] |

Table 1: Average Delay Across All Scenarios

## Experimental Methodology

Based on performance data collected in other studies (Ault and Sharon 2021) (Ghanadbashi et al. 2023), IDQN is the chosen approach to solve the TSC problem. IDQN is an off-policy, model-free RL method that uses a deep neural network (Q-network) to map states to actions. This method incorporates the idea of collecting experience tuples as the agent plays episodes. The experience is used to train a deep neural network randomly shuffled to smooth the training data across many past behaviors (Mnih et al. 2013). The input to the network is the state information, and the output is a Q-value mapped to each action (Asis et al. 2017). Two networks, a trainer and a target are used to enhance model stability. This is because updates are applied to all state-action pairs during training which may include the next state-action pair. To reduce the chances of this happening, the trainer is updated with the target weights less frequently while the target is continuously trained (Fan et al. 2019). An off-policy action selection method known as $\epsilon$-greedy will be used, where $\epsilon$ is the probability the agent will select a random action instead of following the learned policy. Higher values of $\epsilon$ encourage exploration, while lower values encourage exploitation. In the experiments below, $epsilon$ decays linearly from 1.0 to 0.1 at a rate of 0.001 per step (Mnih et al. 2015). For performance evaluation, $epsilon = 0$ The experience replay buffer contains 100,000 elements and training starts when the agent has collected at least 64 experience samples $batch_size = 64$.

### Convolutional IDQN

A convolutional layer was added to the Q-network showing an increase in performance if the state can be represented as an image. The idea behind this is to create an image in which a 2x2 kernel can convolve gathering more meaningful state information if nearby pixels map to state information belonging to the same road (Ault, Hanna, and Sharon 2020). An example of the state space represented as an image is shown in figure 4 where the image displayed on the left is a representation of the traffic shown on the right. Without the convolutional layer at the start of the network, performance was acceptable on the single-agent environment but did not scale well to the 4x4 grid.

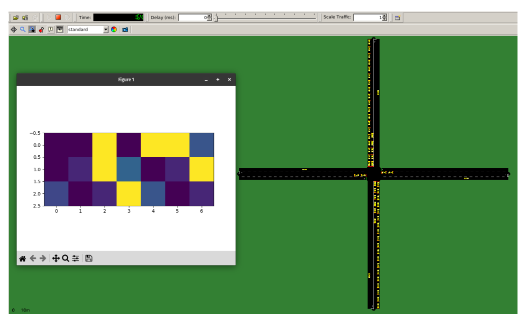

Figure 4: Example of Image-Based State Representation in Convolutional IDQN

After the convolutional layer, 2 fully connected layers are added with 64 units each. The table 2 shows a summary of the network. Huber loss is used as a target for the Q-network to minimize using the Adam optimizer (a variant of stochastic gradient descent) with a learning rate of 0.001. Essentially, Huber loss is a piecewise function combining the resistance to outliers present in absolute loss $L_{abs}(x) = |x|$ and the fast learning (due to its convex nature) of quadratic

[1]Data was only collected in one scenario and may not reflect how the model performs in other scenarios.

[2]Data was extracted from graphs.

loss $L_{quadratic}(x) = x^2$. Huber loss is represented in equation 4 with changes to quadratic and absolute loss to make it differentiable.

$$L_\delta(x) = \begin{cases} \frac{1}{2}x^2, & \text{for } |x| \leq \delta \\ \delta\left(|x| - \frac{1}{2}\delta\right), & \text{for } |x| > \delta \end{cases} \tag{4}$$

| Layer (type) | Param # |
|---|---|
| `Conv2D + LeakyReLU` | 320 |
| `Flatten` | 0 |
| `Dense + LeakyReLU` | 81,984 |
| `Dense + LeakyReLU` | 4,160 |
| `Dense` | 260 |
| **Total trainable params** | 86,724 |

Table 2: Q-Network Model Summary

### Prioritized Experience Replay

Two types of replay buffers were implemented, a traditional replay buffer and a prioritized replay buffer (Schaul et al. 2015). Experiments on agent performance in the 4 x 4 grid scenario were done by testing both traditional replay and prioritized experience replay (PER). PER is chosen to be used in all the agents because it showed better performance than the traditional replay buffer in terms of average cumulative delay during testing ($\epsilon = 0$). In PER, experience tuples are assigned a priority $p$ representing the magnitude of TD-error. During network training, experience $i$ is sampled from the replay buffer with probability $P(i)$ represented in equation 5. The hyperparameter $\alpha$ represents how much prioritization is used with $\alpha = 0$ being equivalent to traditional experience replay. In our experiments, $\alpha = 0.6$.

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{5}$$

To correct bias introduced by the prioritization, importance sampling weights are introduced to adjust the update magnitude of each sample when training the network. Equation 6 shows how these weights are calculated with hyperparameter $\beta$ representing the magnitude of importance sampling correction.

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^\beta \tag{6}$$

During training, $\beta$ starts at 0.000002 and anneals toward 1 at a rate of 0.01% per epoch.

## Performance Metrics

REinforced Signal Control (RESCO) is a testbed and benchmark environment to help judge the performance of RL-based algorithms for TSC. It comes built-in with the SUMO project and will simulate traffic in the same way it was simulated in other benchmark papers (Ault and Sharon 2021). This can be used to compare this project with what has already been done. The metrics that are relevant to this project include total wait time for each vehicle, average delay per vehicle, average number of stops, average queue length, and average trip time (Reza et al. 2023). Studying how the agents respond to injected uncertainty such as an emergency vehicle or increased traffic demands is also an important metric for designing a robust system (Alegre, Bazzan, and da Silva 2020).

## Single Intersection Results

Starting with the simple case, RL methods were applied to a single intersection scenario. Models that performed well on this task were then selected to be used in the 4x4 grid.

### Fixed Time Baseline

To get a baseline to compare the RL approaches, a fixed time agent was implemented using a round robin (RR) approach and cycling the intersection every 50 seconds (Chaudhuri et al. 2021). Figure 5 shows the average cumulative delay for this method. Recall in table 1 the fixed time algorithm had an average delay of 90 seconds. However, better performance is expected in the single intersection scenario, and the 90-second number is an average of 5 different scenarios.

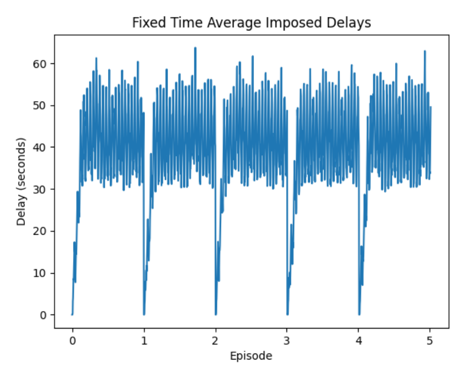


Figure 5: Avg. Delay RR Single Signal

### Convolutional DQN

The convolutional DQN agent was also applied to the single intersection scenario with figure 6 showing the average cumulative delay, and 7 showing the Huber loss function over 5 episodes.

## 4x4 Grid Results

For brevity, the fixed time results applied to the 4x4 grid scenario have been left out of this paper. In summary, the fixed time round-robin approach averages 60 seconds of imposed delays with approximately 100 stopped vehicles in the system at any given time. 16 convolutional IDQN agents were applied to the 4x4 grid, each agent controlling a single traffic signal in the same way as the single intersection experiment. Figure 8 shows average cumulative delay over along with figure 9 showing total stopped vehicles. Figure 10 plots the Huber loss for each agent labeled as in the image 1 above. This experiment required many more episodes than the single intersection due to the independent agent's learning affecting other agents.

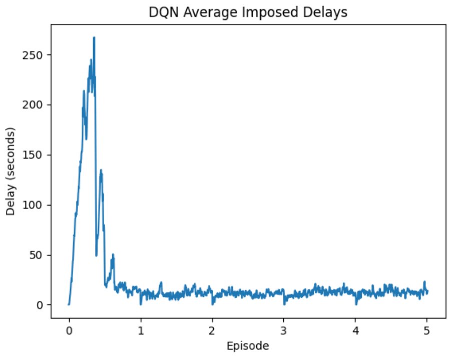


Figure 6: Avg. Delay DQN Single Signal

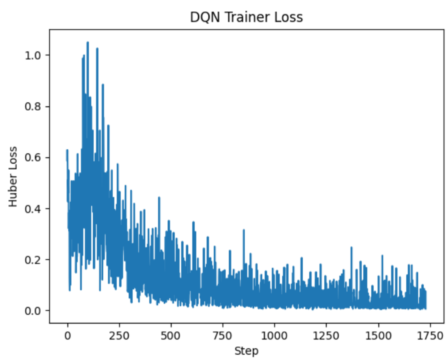


Figure 7: Huber Loss DQN Single Signal

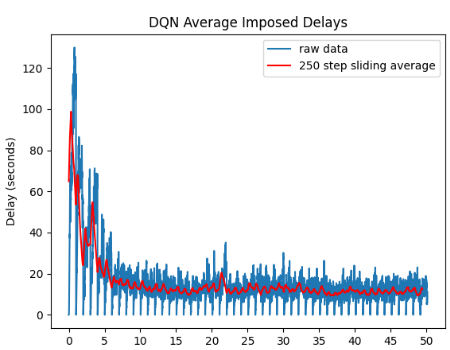


Figure 8: Avg. Delay DQN 4x4

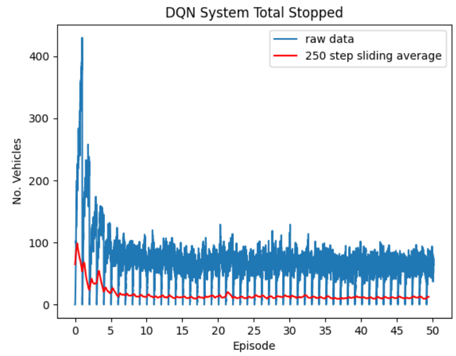


Figure 9: Total Stopped DQN 4x4


Figure 10: Huber Loss DQN 4x4

## Colorado Springs Results

In the Colorado Springs simulation, which encompasses the majority of the downtown area, 203 independent DQN agents are utilized. The average imposed delay per traffic light per vehicle, serving as the reward function, is depicted in Figure 11. The reward function is unchanged from the 4 x 4 grid experiment. Figure 12 illustrates the total number of stopped vehicles within the simulation at each time step. Additionally, Figure 13 displays the Huber loss function's progression over time, with each agent's loss represented by a distinct color. Since there are 203 different plots in figure 13, only information concerning maximum loss over all agents is conveyed.

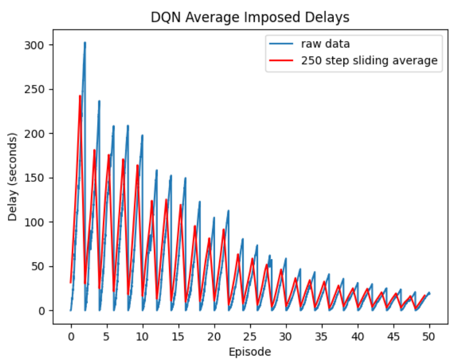


Figure 11: Avg. Delay DQN Colorado Springs

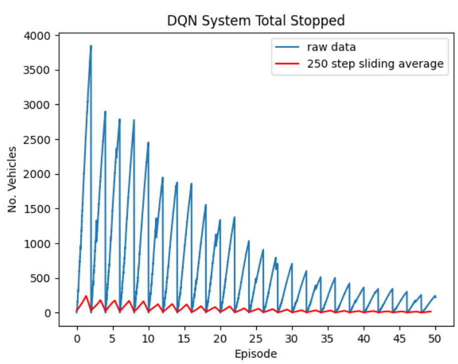


Figure 12: Total Stopped DQN Colorado Springs

On current hardware, the simulation of downtown Colorado Springs demands approximately three hours for every 30-minute episode of actual traffic simulation. Notably, unlike the experiments conducted on the 4 x 4 grid, the Colorado Springs simulation shows an average cumulative delay and number of stopped vehicles resembling a sawtooth wave. This arises from the episode length failing to fully saturate the system. To obtain deeper insights, rerunning the simulation with significantly longer episode durations will provide better conclusions, albeit requiring more than 32GB of RAM. Without achieving system saturation, defining conclusions regarding model performance in all scenarios becomes challenging. Furthermore, increasing the number of episodes for training, or tuning hyperparameters might provide better performance. Considering alternative algorithms, such as Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto, van Hoof, and Meger 2018), could also potentially enhance performance. Overall, the outcomes from simulating Colorado Springs look promising but require further study.

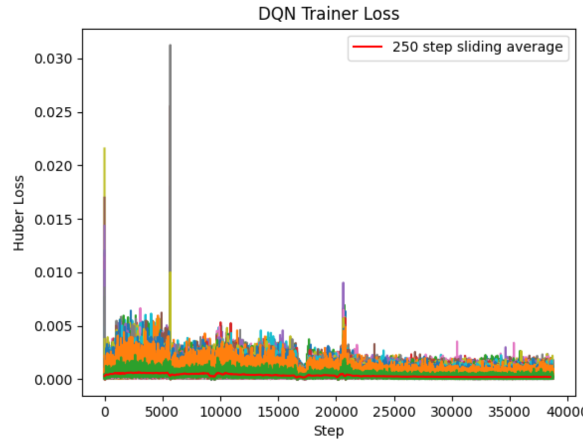


Figure 13: Huber Loss DQN Colorado Springs

## Future Work

The primary focus of the project's future will center around enabling a reinforcement learning agent to manage an operational traffic light. Following discussions with the Colorado Springs signal timing department, tentative approval may be granted for a field test, allowing a reinforcement learning agent to assume control during the traffic light's automatic mode. Exploring this endeavor necessitates specialized hardware capable of interfacing with the traffic light, creating inputs to the existing controller to uphold safety and security protocols.

Moreover, the foundational research projects (referenced as [4] and [5]) instrumental in initializing this project proved invaluable. However, they required debugging and extension to seamlessly integrate with the code from this project, incorporate the requisite features for these experiments, and enhance overall performance. Notably, a pull request has already been merged into the sumo-rl project, with ongoing efforts to contribute further enhancements currently underway.

[4] https://github.com/LucasAlegre/sumo-rl
[5] https://github.com/jault/StateStreetSumo

## Timeline

| Done | Date | Milestone |
|---|---|---|
| ✓ | 2/19 | 4x4 grid environment setup and working |
| ✓ | 2/26 | IDQN agents learning |
| ✓ | 3/4 | Perform experiments and generate data |
| ✓ | 3/18 | Injected uncertainty scenarios |
| ✓ | 3/24 | Colorado Springs experiments |
| ✓ | 4/15 | Data compilation and writing |

Table 3: Project Timeline

## Conclusion

Overall, it was discussed how traffic signal control (TSC) can be formulated in the Markov Decision Process (MDP) paradigm. The state, action, and reward spaces were defined. Based on research done applying the fixed algorithm and reinforcement learning to TSC, an Independent Deep Q-Learning Network approach was used for experiments, aiming to reduce the average cumulative delay $D_t$ to 30.74 seconds or lower. This was achieved using the convolutional IDQN with a cumulative delay of approximately 20 seconds on the 4 x 4 grid experiments. This number did not completely translate to the more realistic experiment simulating Colorado Springs, and we do not provide a number for agent performance when the system is fully saturated with vehicles. Given the partial and non-stationary nature of the TSC problem, its divergence challenges were faced, but overcome. Overall, reinforcement learning could be a feasible approach to traffic signal control.

## References


2022. Advance Basic Traffic Signal Timing and Advance Detection. Website.

Alegre, L. N.; Bazzan, A. L. C.; and da Silva, B. C. 2020. Quantifying the Impact of Non-Stationarity in Reinforcement Learning-Based Traffic Signal Control. *CoRR*, abs/2004.04778.

Asis, K. D.; Hernandez-Garcia, J. F.; Holland, G. Z.; and Sutton, R. S. 2017. Multi-Step Reinforcement Learning: A Unifying Algorithm. *arXiv preprint arXiv:1703.01327*.

Ault, J.; Hanna, J.; and Sharon, G. 2020. 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.

Ault, J.; and Sharon, G. 2021. Reinforcement Learning Benchmarks for Traffic Signal Control. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Chaudhuri, H.; Masti, V.; Veerendranath, V.; and Natarajan, S. 2021. A Comparative Study of Algorithms for Intelligent Traffic Signal Control. *arXiv preprint arXiv:2109.00937*. Available: https://arxiv.org/abs/2109.00937.

Chen, C.; Wei, H.; Xu, N.; Zheng, G.; Yang, M.; Xiong, Y.; Xu, K.; and Li, Z. 2020. Toward A Thousand Lights: Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 3414–3421.

Choi, S.; Yeung, D.-Y.; and Zhang, N. 1999. An environment model for nonstationary reinforcement learning. *Advances in Neural Information Processing Systems*, 12.

Chu, T.; Wang, J.; Codecà, L.; and Li, Z. 2019. Multi-Agent Deep Reinforcement Learning for Large-scale Traffic Signal Control. *CoRR*, abs/1903.04527.

de Almeida, V. N.; Bazzan, A. L. C.; and Abdoos, M. 2022. Multiagent Reinforcement Learning for Traffic Signal Control: a k-Nearest Neighbors Based Approach. In *Twelfth International Workshop on Agents in Traffic and Transportation, co-located with the 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence (IJCAI-ECAI 2022)*, volume 3173 of *CEUR Workshop Proceedings*. Vienna, Austria: CEUR-WS.org.

Fan, J.; Wang, Z.; Xie, Y.; and Yang, Z. 2019. A Theoretical Analysis of Deep Q-Learning. *arXiv:1901.00137*.

Federal Highway Administration. 2023. Manual on Uniform Traffic Control Devices for Streets and Highways. Federal Highway Administration.

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 1582–1591.

Ghanadbashi, S.; Safavifar, Z.; Taebi, F.; and Golpayegani, F. 2023. Handling uncertainty in self-adaptive systems: an ontology-based reinforcement learning model. *Journal of Reliable Intelligent Environments*.

Hafiz, A. M.; and Bhat, G. M. 2020. Deep Q-Network Based Multi-agent Reinforcement Learning with Binary Action Agents. arXiv:2008.04109.

Lee, K. M.; Ganapathi Subramanian, S.; and Crowley, M. 2022. Investigation of independent reinforcement learning algorithms in multi-agent environments. *Frontiers in Artificial Intelligence*, 5.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5602.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Reza, S.; Ferreira, M. C.; Machado, J. J. M.; and Tavares, J. M. R. S. 2023. A citywide TD-learning based intelligent traffic signal control for autonomous vehicles: Performance evaluation using SUMO. *Expert Systems*. First published: 05 April 2023.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952*.

Terry, J.; Black, B.; Grammel, N.; Jayakumar, M.; Hari, A.; Sullivan, R.; Santos, L. S.; Dieffendahl, C.; Horsch, C.;

Perez-Vicente, R.; et al. 2021. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 15032–15043.