CSE 291 Project Final Report: Constraints in Multi-Agent Traffic Signal Control (Con-MATSCo)

Rohin Garg
Computer Science, UCSD
rgarg@ucsd.edu

Mayank Sharan
Computer Science, UCSD
msharan@ucsd.edu

Ulyana Tkachenko
Computer Science, UCSD
ulya.tkach@gmail.com

1 Introduction

1.1 Problem Introduction

Population growth and densification of urban neighbourhoods is leading to traffic increasingly becoming a blocking problem in real life. Adaptive traffic signal control (ATSC) solutions attempt to tackle road congestion by modeling stoplight behavior policies based on real time traffic data. Initial works relied on heuristics and domain specific information to create stoplight policies, however, collecting such information for larger road maps is time exhaustive, expensive and lacks feasibility.

Recently, reinforcement learning (RL) has been applied to the task. RL algorithms do not rely on heuristics and are capable of learning optimal policies by directly fitting a parametric model on the input space, making them provably useful in the ATSC domain. However, these policies by default do not take into account real world constraints to ensure tractability and safety.

Designing agents with safe policies would make them deployable for traffic signal control in the real world. Our work is focused on incorporating both hard and soft constraints into the learning of the RL agent to ensure safety and allow configurability. The project is being maintained on **github**[1].

1.2 State-of-the-Art

Single traffic intersections have seen many implementations of RL algorithms for traffic control from Actor-Critic [1] to Deep Q-learning [2], however due to a scalability constraint the size of the road maps these agents could inform were extremely limited (i.e. single street light).

More recent decentralized multi-agent RL studies have attempted to tackle this issue implementing algorithms like LR IQR [3] and MARL [4] with neighborhood communication for partial observability, but these algorithms were tested on small datasets. MPLight [5] addressed this issue in a large-scale traffic signal control scenario with over 2500 traffic lights. CoLight [6] works in a similar direction as MPLight coordinating multiple RL agents accross multiple intersections.

These works have used Simulation of Urban MObility (SUMO) [7] to perform traffic simulations and benchmark performance. CityFlow [8] is a second, more recent traffic simulator that claims to be better adapted for large scale traffic modeling however it is not as widely adapted in research at this time. Despite the novel improvements to the RL agents, none of the algorithms have a specific focus on constraining agent behavior for safety or track metrics to measure safety performance.

1.3 Methodology

The work primarily focuses on leveraging the superlative performance of RL agents in the ATSC task and improving it to allow for deployability in the real work. Such agents needs to adhere to safety

[1] https://github.com/utkachenko/Con-MATSCo

Preprint. Under review.

parameters, guidelines and laws of the road. These requirements necessitate that constraints need to be embedded in agent behavior.

This paper presents traffic signal control which is efficient, generalizable, adaptive, safe and realistic to handle the increasing burden on the roads in the following ways:

This paper aims to answer the following research questions pertaining to traffic signal control which is efficient, generalizable, adaptive, safe and realistic to handle the increasing burden on the roads.

- **RQ1:** How to define metrics and formulate the problem for constraint inclusion?
- **RQ2:** How does the SoTA perform with respect to constraint violation metrics?
- **RQ3:** Which methods do / do not work for incorporating constraints in a multi-agent setting?
- **RQ4:** Which methods are more suitable for soft/hard constraints?
- **RQ5:** How does constraint inclusion impact the performance of the agent?

2 Related Work

AI Safety Performance of RL agents is often decreased when incorporating constraints in their behavior. Open AI's SafetyGym [9] is a toolkit environment for developing RL algorithms that adhere to specific safety constrains during training and measuring the success of these algorithms throughout the learning process. The main contribution of SafetyGym is to promote research into developing standardized 'safety metrics' for agent performance relative to the cost functions the agent is constrained by. For our model, we plan to expand the implementation of constraints past utilizing an auxiliary cost function and therefore require more robust metrics to measure AI performance beyond the scope of SafetyGym.

Safe RL For RL agents to effectively explore the action space, agents risk picking actions that lead to domain specific unsafe situations. Avoiding reaching these states during training thus makes 'safe exploration' another major research topic in RL. DeepMind's recent research [10] focuses on using reward modeling to predict the reward of these actions without necessitating the agent directly interacts with the environment or enters an unsafe state space. Synthesizing query reward is well applicable to huge actions and state spaces, where all unsafe states cannot be explicitly defined, however, with such a small subset of actions and well defined unsafe situations in traffic control this is less of a focal issue.

Constrained Optimization There is limited work on constrained policy search for deep RL. [11] makes assumptions for a robotic manipulation environment that prevents their method from being used across domains. Results on other real world applications such as traffic signal control are not available. But we plan on using the penalty based rewards given in [11] in our experiments. It might be possible to make existing algorithms more efficient when applied to specific applications. [12] extend Trust Region Policy Optimization algorithm [13] for Constrained MDPs. [14] provide a model based controller based on the constraints. This controller can be used on a model-free policy learning algorithm to learn within states that don't violate constraints with a high probability.

3 Methodology

3.1 Problem Definition

The RL approach we are using as the basis to build our solution upon is CoLight [6]. In this approach each intersection behaves as an individual RL agent but enables cooperation amongst agents by providing weighted state inputs, using attention, from neighboring intersections.

- **Inputs**:
  - Roadnet file: This file defines the road network representing intersections and roads as nodes and edges of a graph. A sample is shown in Figure 1.
  - Flow file: This file define the attributes associated with each vehicle that is a part of the simulation.

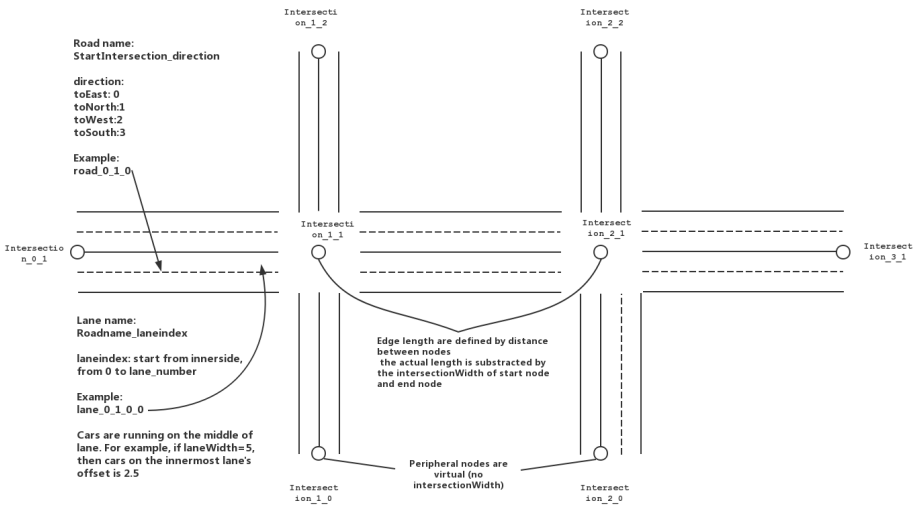

Figure 1: Roadnet Illustration for a 1x2 intersection grid

- **Outputs**: The state representation $s_{t,i}$ at every time step $0 \leq t < 3600, t \in T$ for each intersection $i \in I$ for the complete simulation for a total of $I \times T$ states.
- **State space**: The state, $s_{t,i} \in S$ at time t each intersection $i \in I$ in the network is represented as a dictionary containing the following attributes, assuming the number of lights and number of lanes at an intersection are k and l:
  - `current_phase`: The state of lights at the intersection. This is a categorical vector of length k.
  - `time_this_phase`: The time steps for which the intersection has been in the current phase where t is a whole number s.t. $0 \leq t < 3600$.
  - `lane_num_vehicle`: Number of vehicles per each lane, a length l vector.
  - `lane_num_vehicle_been_stopped_thres1`: Number of vehicles per lane waiting for more than 1s, a vector of length l.
  - `adjacency_matrix`: A vector of length 5 containing indices of the intersections adjoining the current intersection.
  - `adjacency_matrix_lane`: For each lane contains a map of all input lanes and output lanes associated with it. This is a matrix of size $2l$ x 5.
  - `lane_wait`: A vector of length k containing average wait times per signal for which it stays red.

  The combination is a state of length $2k + 12l + 6$
- **Reward function**: The baseline approach uses a weighted negative sum of number of vehicles waiting per lane for more than 1 seconds as the reward function for each intersection. This penalizes any action that would increase the overall wait time for the vehicles.
- **Actions**: The action, $a \in A$ at every intersection i either maintains the current phase of lights or switches to a different phase. $|A| = 5$ as there are 5 phases in case of a 3 light, 4 way intersection. The phases are:
  - North straight and South straight allowed
  - North left and South left allowed
  - East straight and West straight allowed
  - East left and West left allowed
  - All lights yellow

3.2 Mathematical Formulation

We will represent the number of intersections as n_{int} and the number of time steps in a simulation as T.

State State is as described in the previous section a vector of length $2k + 12l + 6$ containing categorical, integral and continuous values. We have a state for each intersection i at time t represented as s_t^i. We will describe the components of the state at time t for intersection i as follows -

- ϕ_t^i : `current_phase`
- τ_t^i : `time_this_phase`
- δ_t^i : `lane_num_vehicle_been_stopped_thres1`
- ν_t^i : `lane_wait`

Action Action is an integer from a set of cardinality 5. There is an action corresponding to each intersection i at each time t represented as a_t^i.

Reward The baseline approach has the reward function for each intersection i and time t as

$$r_t^i = -0.25 \sum_{lane=0}^{l} \delta_{t,lane}^i \tag{1}$$

This penalizes any action that would increase the wait time for the vehicles overall.

Constraints We have worked on incorporating 2 constraints

- **Minimum Signal Switching Time Constraint**: The SoTA policy is susceptible to learning actions in scenarios where a light is turned to green for as short a time as 1 second. While this may work in a simulation it creates high risk scenarios where accidents can occur in the real world. This can be represented as

  $$\phi_{t+1}^i \neq \phi_t^i \rightarrow \tau_t^i \geq T_{switch} \tag{2}$$

  where T_{switch} is the minimum time required between signal switches. This can be used as a soft or a hard constraint.
- **Signal Switching Fairness Constraint**: The RL agent is looking to minimize the average travel time which leads to learned behavior where certain intersections might not switch on certain signals for a long time as long as a significantly higher number of cars is waiting at the other signals. This is unfair to the person who would be waiting on that signal for extremely long periods of time. This can be shown for a given intersection as:

  $$\max_{lane} \nu_{t,lane}^i - \min_{lane} \nu_{t,lane}^i \leq T_{gap} \tag{3}$$

  where T_{gap} is the maximum allowed gap between average signal switch times for signals on the same intersection

3.3 Methodology Description

3.3.1 Hard Constraints

Hard constraint modeling is relatively straightforward and can be done by restricting the action space or through rules in the policy execution. This was relatively less interesting as compared to Soft constraint modeling as this gives very little leeway to the policy around the constraints and becomes similar to heuristic based policies in practice. An interesting way to consider modeling hard constraints is learning the policy by modeling it as a soft constraint and then limiting the action space at the time of execution. This would allow for the policy to have a bigger space to learn in and satisfy the hard constraints. We will cover results for hard constraint modeling for the rule based agents in the experiments section.

3.3.2 Soft Constraints

We explored a few different methods for incorporating soft constraints into the policy.

Reward Shaping Penalty terms were added to the reward function to represent constraint violations to train the agent to follow them. The mathematical formulations of the same are -

- **Minimum Signal Switching Time Constraint**:

$$r_t^i = -0.25 \sum_{lane=0}^{l} \delta_{t,lane}^i - 3 * \mathbb{I}[\phi_{t+1}^i \neq \phi_t^i, \tau_t^i < T_{switch}] \tag{4}$$

- **Signal Switching Fairness Constraint**:

$$r_t^i = -0.25 \sum_{lane=0}^{l} \delta_{t,lane}^i - \max_{\lambda_1, \lambda_2 \in \nu_t^i} \frac{1}{2}(D_{KL}(X_1||X_2) + D_{KL}(X_2||X_1)) \tag{5}$$

  where $X_1 \sim exp(\frac{1}{\lambda_1})$ and $X_2 \sim exp(\frac{1}{\lambda_2})$ modeling the wait time for each lane using an exponential distribution and D_{KL} represents the KL-divergence distance.

Adaptive Q-Learning We adjusted the target q-values that the q-network trains to with penalties and rewards for violating and adhering to the constraints respectively. This was only done for the Minimum Signal Switching Time constraint due to time limitations. The training target q value update is set as

$$q_t^i = r_t^i + \gamma * {q_{target}}_t^i + 0.1 * \mathbb{I}[\phi_{t+1}^i \neq \phi_t^i] * sgn(\tau_t^i - T_{switch}) \tag{6}$$

Constraint Policy Optimization, or Trust Region Optimization method for Constrained MDPs. *Constrained Markov Decision Processes* as defined by [12]: An MDP with auxiliary cost functions for each constraint $C : C_1, C_2...C_m$, where $C_i : S \times A \times S \to \mathbb{R}$, and limits for each costs $d_1, ...d_m$. $J_{C_i}(\pi)$ denotes the expected discounted *cost* of policy π, and thus the set of feasible stationary policies in a C-MDP ($\prod_C$) are:

$$\prod_C = \{\pi \in \prod : \forall i, J_{C_i}(\pi) \leq d_i\}$$

The new update problem for C-MDPs then becomes:

$$\pi_{k+1} = \text{argmax}_\theta J(\pi)$$

such that

$$J_{C_i(\pi)} \leq d_i \forall i = 1...m; D(\pi, \pi_k) \leq \delta$$

where D is some distance measure and $\delta > 0$ is a step size for policy gradient update.

Inspired by TRPO, the authors of CPO [12] propose the following updates for Constraint-MDPs:

$$\pi_{k+1} = \text{argmax}_{\pi \in \prod_\theta} \mathbb{E}[A^{\pi_k}(s, a)]$$

s.t.

$$J_{C_i}(\pi_k) + \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_k}, a \sim \pi}[A_{C_i}^{\pi_k}(s, a)] \leq d_i \forall i; \bar{D}_{KL}(\pi||\pi_k) \leq \delta$$

In the above update rule, the variables are inherited from the original TRPO update rule [13]. Since this is a trust region method, it inherits the performance guarantees of TRPO. The authors further prove a performance guarantee for an approximate satisfaction of constraints.

The experiments done by the authors are done on Mujoco environment for simple single-agent tasks like point-gather that require very low capacity neural networks to learn the value function and the policy. We were able to use CPO on the point-gather environment and replicate the results of the paper. We tried to use the same learning algorithm for Cityflow dataset using the code provided by CoLight repository. But our models were not able to learn the policy effectively. This could be due to reasons such as insufficient model capacity and errors in our CoLight code modification for CPO (since we had to adapt across deep learning frameworks from tensorflow [CoLight] to pytorch [CPO]), incorrect sampling as required by TRPO based methods etc.

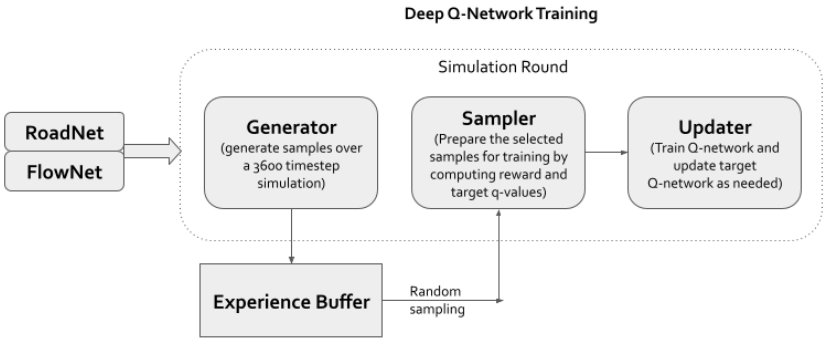

Figure 2: Setup for training the multi-agent ATSC policy

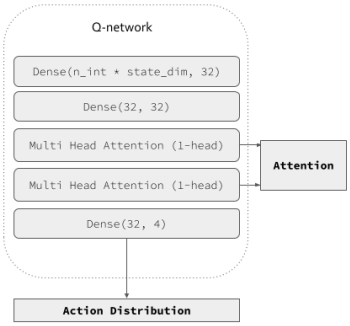

Figure 3: Q-network architecture

3.3.3 Method Overview

Figure 2 show the setup used to train the agents. The simulation rounds run sequentially to generate experiences which are randomly sampled to generate training samples for the Q-network. The sampler is where the modifications are made for incorporating soft constraints as the reward and the target q values are both computed there.

A few modifications from our end in the SoTA approach are -

- We observed that the generator was using the base q network to generate samples making the approach on-policy. This explained the high variance we were seeing in the performance across simulation rounds. We changed this to off-policy learning by using the target q-network.
- We ran all agents with minimum action time set to 1s rather than the 10s used by the SoTA. This was to allow for the agent learning a policy that was not artificially limited by unexplained constraints.
- We rewrote the code for the SoTA from scratch fixing some bugs along the way

3.4 Algorithm Description

Algorithm 1 Overall Algorithm Pipeline

while $c \leq N_{rounds}$ **do**
  Generate($c, T = 3600$) ▷ Generate samples for round c for T timesteps
  Sample_Construct(c) ▷ Construct $[S, A, R, Target - Q]$ for selected samples
  Update_Network() ▷ Update Q network
end while

function Sample_Construct(c):
while $i \leq n_{int}$ **do**
  while $t \leq T$ **do**
    $r_t^i \leftarrow$ calculate_reward(s_t^i, a_t^i) ▷ Calculates reward (4) or (5)
    $q_{target}{}_t^i \leftarrow$ q_target_computation (s_t^i, a_t^i, r_t^i) ▷ Calculates target-q value
  end while
end while

The above pseudocode shows the high level flow of the training algorithm. The reward shaping and adaptive q-learning changes were made in `calculate_reward` and `q_target_computation` functions respectively.

4 Experiments

4.1 Tasks and Evaluation

Datasets We are working with the following datasets -

- Synthetic 3x3: This is a synthetically generated dataset with 9 intersections arranged in a 3x3 grid.
- Synthetic 6x6: This is a synthetically generated dataset with 36 intersections arranged in a 6x6 grid.
- NewYork 16x3: This is a real world dataset for 48 intersections from Manhattan.

The synthetic 3x3 grid was used for setup and experimentation. Synthetic 6x6 grid has been used for evaluation and hyperparameter tuning of the agents with and without constraint inclusion components due to its balance of reasonably large number of agents and manageable run times. The NewYork 16x3 grid was used to test the agent's performance on real-world data at the end of the experiment.

Baselines Baseline SOTA model MPLight [5] proved to be an issue to run as based on correspondence with the author that code could not be run without extensive refactoring. Instead, well maintained model CoLight [6] was used as the baseline RL model.

We used a couple of rule/heuristics based agents as baselines

- **FixedTime** is a policy where after a fixed amount of time the lights switch cyclically. This is very widely used in the real world to control traffic lights
- **MaxPressure** is a greedy policy that chooses the phase that corresponds to the maximum pressure - a pre-defined metric about upstream and downstream car queue length. This method finds its origins in control theory.

It is possible that heuristics based approaches might be better at incorporating constraints so we evaluate our approach on not only the model but both baselines as well.

Metrics CoLight[6] reported average travel time in seconds as the evaluation metric. This is the central metric to traffic signal control.

We have designed additional metrics designed to quantify the extent of constraint violation for minimal signal switch cost and the signal fairness cost corresponding to the constraints.

- **Min Signal Switch Cost**: Say,

  $n_{violations} = \sum_{t=0}^{T} \sum_{i=0}^{n_{int}} \mathbb{I}[\phi_{t+1}^i \neq \phi_t^i, \tau_t^i < T_{switch}]$

  $n_{switches} = \sum_{t=0}^{T} \sum_{i=0}^{n_{int}} \mathbb{I}[\phi_{t+1}^i \neq \phi_t^i]$

  then the cost is,

  $$\frac{1}{2}\left(\frac{\sum_{switches} max(0, T_{switch} - \tau_t^i)^2)}{n_{violations} * T_{switch}^2} + \frac{n_{violations}}{n_{switches}}\right) \tag{7}$$

  This cost ranges from 0 to 1 irrespective of the T_{switch} set allowing this to be used across various threshold values. This also includes an equally weighted component of ratio of violations to ensure that few violations are not penalized as harshly as regular violations.

- **Signal Switching Fairness Cost**

  $$\mathbb{E}[\max_{\lambda_1, \lambda_2 \in \nu_t^i} |\lambda_1 - \lambda_2|] \tag{8}$$

  The mean absolute error captures the fairness violation while being easily interpretable as seconds of gap.

4.2 Implementation Details

CoLight model, FixedTime, MaxPressure baseline implementations and the constraint inclusion methods were run using the following setup -

- The code for the CoLight agent is written using tensorflow. This uses version 1.14. The code was highly convoluted and riddled with bugs. We rewrote the code from scratch to be able to run experiments. The baselines are written in plain python.
- The simulation environment used is CityFlow. [8]
- We ran the code on our personal machines with a typical configuration of a 15" Macbook Pro running with CPU and 16GB RAM.
- The heuristic baselines do not require any training time and run the simulation in about 2 minutes each for the synthetic 6x6 dataset. The CoLight model requires $\tilde{4}$ hours to train on the 6x6 synthetic dataset for 50 rounds of simulation.
- The simulation is run for 3600 seconds.
- The seeds for all model runs were fixed to ensure parity in results across runs.
- Hyperparameter tuning was done by intuitive search. We started with a few candidate values based on our understanding of the model and the setup and selected the value that generated the best results
- Each model was run for 50 rounds each of which includes 1 complete simulation. The ideal run would have been for 100 rounds as that was used by the SoTA paper and the models were still converging but we chose 50 to adhere to time limitations.

4.3 Experiment Results

Quantitative Results The models were tested on the synthetic 6x6 dataset and NewYork 16x3 real dataset and the average travel time and the constraint metric for each model are show below. Reward Shaping and Adaptive Q-Learning refer to soft constraint inclusion modifications made to CoLight. FixedTime refers to the default agent while FixedTime-Hard refers to the agent where the time between signal switches was set to T_{switch} to ensure that the constraint is followed. We evaluated for T_{switch} set to 15 seconds. The best performance and the second best in case the best is from a rule based model are marked in bold.

The lower the average time is better and the same follows for constraint metrics as well.

Table 1: Evaluation for Signal Switch Constraint Metrics

	Synthetic 6x6		NewYork 16x3	
Constraint Metric	Avg Duration	Constraint Metric	Avg Duration	Constraint Metric
FixedTime	532.69 ± 0.0	0.722 ± 0.0	1744.67 ± 0.0	0.722 ± 0.0
FixedTime-Hard	517.35 ± 0.0	**0.0** ± **0.0**	2302.67 ± 0.0	**0.0** ± **0.0**
MaxPressure	351.16 ± 0.0	0.66 ± 0.0	1836.92 ± 0.0	0.723 ± 0.0
CoLight	**215.31** ± **38.47**	0.48 ± 0.06	1662.58 ± 32.94	0.57 ± 0.02
Reward Shaping	573.64 ± 11.95	0.63 ± 0.07	1771.68 ± 80.21	0.64 ± 0.02
Adaptive Q Learning	296.87 ± 28.34	**0.29** ± **0.03**	1758.33 ± 120.6	0.52 ± 0.1

Table 2: Evaluation for Signal Fairness Constraint Metrics

	Synthetic 6x6		NewYork 16x3	
Constraint Metric	Avg Duration	Constraint Metric	Avg Duration	Constraint Metric
FixedTime	532.69 ± 0.0	0.0 ± 0.0	1744.67 ± 0.0	0.0 ± 0.0
CoLight	**309.44** ± **42.87**	6.88 ± 0.39	**1512.81** ± **58.18**	8.98 ± 0.28
Reward Shaping	377.56 ± 77.97	**6.14** ± **0.94**	1561.65 ± 24.53	**6.87** ± **0.37**

As we can see there is a clear established trade-off between following constraints and minimising average travel time. It is notable that the approaches attempting to incorporate constraints manage to perform comparably to the baseline while showing significant improvement in the constraint metrics.

We can see that for the signal switch constraint metrics the reward shaping method is worse than the baseline on both counts, average travel time and the constraint metric. This is likely due to improper penalty formulation or requirement of better hyperparameter tuning. It does not rule out reward shaping as a meaningful constraint inclusion method as we can clearly see that reward shaping does well to improve on the signal fairness constraint.

(a) t=1243 (b) t=1244 (c) t=1245 (d) t=1246 (e) t=1247

Figure 4: Dangerous intersection behavior for baseline CoLight policy

Qualitative Results Figure 4 shows snapshots of an intersection from the GUI simulation of the policy learned by CoLight. As can be noticed here the left turn light coming in from the north intersection gets switched on stays on for 3 seconds and is then turned of. This is highly dangerous as a car that started moving with this green would be still moving through the intersection when other lights turn on leading to accident risk.

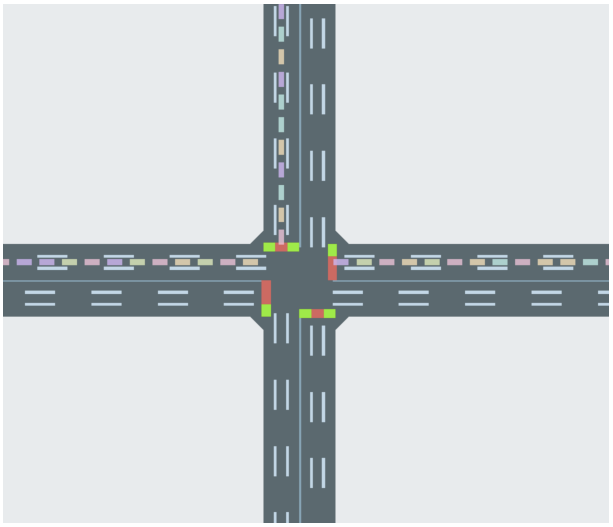

Figure 5: Illustration of a logjam due to light not being turned green for long periods of time

An interesting behavior we came across while analyzing the output policies is that sometimes the agent falls in a pattern where a few lights are not switched on, even though there is no reason for them to not be, for extremely long periods of time leading to logjams. This leads to degraded performance increasing variance during training and problematic policy behaviors. This is an interesting issue to look at resolving through constraints as well.

Unfortunately, these are the only GUI simulations we could include in the report given the fact that our constraints are time based it will take a large number of screenshots to show different behavior.

5 Conclusion and Discussion

5.1 Conclusion

In this project, we looked at approaches to incorporate constraints in multi agent traffic signal control. As expected there was a trade-off discovered between the average travel time and adherence to the constraint. Although, it is interesting to note that vehicles in real world do not behave as smoothly as the ones in simulation and the requirement of constraints is predicated on the same. This could also adjust for gaps in the performance as the constrained policies are better adjusted for human behavior.

Our contributions were designing constraint metrics, implementing reward shaping and developing a novel method for incorporation of soft constraints in adaptive q-learning that from early indications seems to perform quite well.

5.2 Discussion

The intent of constraint multi agent traffic signal control is to enable safe real world deployment of RL models. These approaches warrant further exploration with more complex constraints and with experiments to incorporate multiple constraints in a single policy. There are numerous variations of experiments to be run with the methods we introduce in the project which we were not able to explore fully due to time constraints. Another, limitation that would warrant further exploration is that the approaches we present require domain knowledge to effectively design measurement metrics and constraint inclusion components which limits transferability across tasks and domains. A method like CPO that is more mathematically general or mathematical categorisation of constraint inclusion options with reward shaping and adaptive q-learning would go a long way in ensuring generalizable constraint inclusion approaches.

References

[1] Silvia Richter, Douglas Aberdeen, and Jin Yu. Natural actor-critic for road traffic optimisation. *Advances in neural information processing systems*, 19:1169–1176, 2006.

[2] Wade Genders and Saiedeh Razavi. Evaluating reinforcement learning state representations for adaptive traffic signal control. *Procedia computer science*, 130:26–33, 2018.

[3] Tianshu Chu, Shuhui Qu, and Jie Wang. Large-scale traffic grid signal control with regional reinforcement learning. In *2016 american control conference (acc)*, pages 815–820. IEEE, 2016.

[4] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1086–1095, 2019.

[5] C. Chen. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. *AAAI*, 34(4):3414–3421, 2020.

[6] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. Colight. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, Nov 2019.

[7] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo-simulation of urban mobility. *International journal on advances in systems and measurements*, 5(3&4), 2012.

[8] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David C. Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. *CoRR*, abs/1903.09254, 2019.

[9] Joshua Achiam and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. 2019.

[10] Siddharth Reddy, Anca D. Dragan, Sergey Levine, Shane Legg, and Jan Leike. Learning human objectives by evaluating hypothetical behavior. *CoRR*, abs/1912.05652, 2019.

[11] David Held, Zoe McCarthy, Michael Zhang, Fred Shentu, and Pieter Abbeel. Probabilistically safe policy transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5798–5805. IEEE, 2017.

[12] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

[13] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[14] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.