All of the (Traffic) Lights – Q-Learning for Urban Traffic Optimization

67842 Introduction to Artificial Intelligence – Final Project

Dan Amir
dan.amir@mail.huji.ac.il

Netai Benaim
netai.benaim@mail.huji.ac.il

Omer Dolev
omer.dolev1@mail.huji.ac.il

May Yaaron
may.yaaron@mail.huji.ac.il

***Abstract*—Efficient traffic light signal timing is a key challenge in urban road settings. Currently, most traffic light controllers use fixed timings or basic heuristics to distribute right-of-way at intersections. We propose a reinforcement-learning based multi agent traffic light system, modeled on a compact state space, aimed at optimizing overall commute time and minimizing car accidents. Our agents were trained on a basic 8-intersection setting using an urban traffic simulator and compared against fixed-timing and heuristic-based traffic light agents, outperforming both on waiting time, congestion and car accidents optimization targets. Additionally, we discuss potential improvements to our model using a more elaborate feature space and increased coordination between traffic light agents.**

I. Introduction

Society relies heavily on transportation systems. With people often not residing within walking distance of their workplace the number of commuters is ever increasing. A 2015 report [2] claims urban commuters waste about 42 hours a year stuck in traffic jams, and data from 2017 shows that in metropolitans like LA and New York the average values are 102 and 91 hours respectively.
Aside of the general annoyance of being stuck and wasting time, there are other consequences for heavy traffic such as loss of working hours and substantial CO2 emissions. This problem can generally be addressed by two types of solutions. The first is expansion of infrastructure to accommodate increased traffic – that is, to build more roads. The second is an increase in efficiency of existing infrastructures, mainly making the traffic routing systems (traffic lights, GPSs) more aware of the traffic and add the ability for better online planning of traffic routing.
A recent article [3] has laid out the ways in which Artificial Intelligence can be and already is harnessed to deal with the problem. Alongside the numerous companies that already started gathering traffic data and test running AI controlled traffic systems, there is a significant body of academic work that has been exploring the field as far back as the 90s. The article mentions companies that have tried to create AI systems based on data collected from road traffic cameras and other sensors. A recent paper [4] has shown that interestingly, the types of sensors used to gather data may not affect the learning outcomes significantly and that AI controlled traffic light systems may be trained with a fairly abstract representation. Consequentially, we saw fit to use a traffic simulation in our project.
Our goal for this project is to build an efficient traffic lights system using AI. Said system should include several traffic lights (multi-agent system) in spatial proximity, such that the actions of one traffic light will affect the stream of vehicles to the others.
We wish to build an autonomous traffic control system that relies only on external traffic data (meaning the system doesnt know the routes and schedules of the commuters). Our idea is to create a multi agent system where each agent is an independent traffic light that knows nothing about the other agents behavior and observations after deployment. We believe our simple approach system could prove a viable and less error-prone traffic system for real-world implementation. In order to allow some coordination between the traffic lights we will share limited information between the agents during their training phase.
The general aim of the system is to minimize the total vehicles waiting time – that is, minimize the effects of the urban traffic on the optimal commute time (i.e. on an empty road). A secondary aim is to eliminate any urban traffic-related risks to drivers and passengers that may occur due to traffic light behavior.

II. Approach and Methods

A. Simulator and Learning Framework

We chose to use SUMO [5] – an open source traffic simulator that is often used in research in the transport optimization domain. Since the traffic environment is dynamic, involves an infinite time horizon and has a relatively rich state space and a very simple action space, we decided to use reinforcement learning to tackle our problem.

B. Environment Formulation

In our model, an agent controls a single traffic light with the goal of minimizing the time takes to vehicles to pass the junction while keeping the drivers safe. Using an MDP framework, our model was formulated as follows.

Action: The logic of standard traffic is usually modeled using phases. Each phase represents a specific **valid** combination of green-yellow-red lights for all different lanes. A simple traffic light is controlled by setting a fixed timing for a cyclic transition between those phases. Two reasonable approaches for formulating the action space (and by so also the time resolution of the MDP) are either to represent full cycle of

phases timings as a single action and give the agent control once in a cycle, or to represent the phase choice in each simulation time-step as a separate action.
Although the first approach can result in more stable training, we believed that it will significantly limit the agents freedom to generate efficient and non trivial behaviors based on real time data. Therefore, we chose the second approach and experimented with two distinct implementations of the action space – in the first the agent chooses the specific phase for each step; in the second the agent chooses whether or not to move to the next phase in the cycle for each time-step.
In our experiments, we were not successful in finding hyper parameters for the binary decision space, even for a single agent environment; therefore, we focused our efforts on the phase choice approach.

State: We assume that the traffic light has access to information regarding the velocity and location of any vehicle within some predetermined radius from the junction (to be referred to from now on as the junction polygon), as well as to internal information regarding the amount of time spent since its last phase change. As a complete representation of all parameters – combinations of different vehicles locations and speeds and internal phase durations – would yield an enormous state space, we decided to use Approximate Q-Learning with a linear function over a compact feature representation of the (state, action) pairs. Our final feature vector is composed of three subparts: three phase features, four green light lanes features, and four red light lanes features.
The phase features consist of an indicator to whether the chosen action will keep the traffic light at the same phase; the duration of the current phase after performing the action; and an indicator of whether the chosen phase is a yellow light phase (e.g. transition between green phases).
The lane features, which are computed for all incoming lanes, consist of the location of the closest car to the junction on the lane; the number of cars on the lane in the junction polygon; the vehicles average speed; the vehicles average waiting time, which is computed as the time passed since the car moved faster than 0.1m/s. In order to obtain the four red lanes features we sum those values over all incoming lanes that will have red or yellow light after performing an action. The green light lanes features are computed from the rest of the lanes. Since the original features resulted in divergence and exploding weights, we normalized each feature to a 0-1 scale.
We had also experimented with more expressive feature representations, where the features were included separately for each combination of lane-action; however, we found those to result in unstable training. Since the rich representation resulted in feature vector of length 180, we assume that the reason for the results lies in the difficulty to generalize from such a representation, as the agent has to learn the importance of each feature for each lane separately, and the inability of the linear function to condition the weight that said features are given on the traffic light state for the specific lane and action combination.

Reward: We used the average time passed since a car arrived to the junction polygon over all cars in incoming lanes, as a momentary approximate measure of the delay effect of the traffic light behavior. In order to better regulate the agent against risky behavior, we added the following weighted terms to the total reward - flickering punishment and accident punishment. The flickering term is set to 1 when the traffic light changes its phase and 0 otherwise. The accident term is set to the number of accidents which occurred in the junction polygon during the last time-step (both terms were inspired by previous works on the subject [6]). Therefore our final reward was set to:

$$R(j) = -(w + 0.1f + 2a)$$

Where $R(j)$ is the total reward for a junction j, w is the accumulated waiting time, f is the flicker factor and a is the penalty for each accident occurring withing the junction polygon.
Additionally, we experimented with two other reward terms - the average difference between cars speed and maximal allowed speed (which we named the delay factor) and the number of emergency car stops in the junction. We decided to exclude these from the final total reward, since the delay factor didnt seem to affect the experiment results and the emergency brake factor resulted in unstable training. The final reward used for training the agent is a convex sum of the local junction reward and the rewards of the adjacent junctions. The amount of contribution of adjacent junction rewards to the final reward is controlled by the hyper parameter r_{weight}, such that the total weighted reward is:

$$R_t(j) = r_{weight} * R(j) + (1 - r_{weight}) * \frac{1}{N} \sum_{k}^{N} R(j_k)$$

Where k is the adjacent junction index and N is the number of such junctions.
Although this reward sharing scheme with no state and internal agent state information sharing can interfere with the Markovian structure of the domain, we assumed that the combination of local state observation and global reward will allow the agents to balance their outgoing traffic volumes in an efficient manner.

III. Experiments

A. Experimental Setup

In order to simulate a realistic urban traffic situation with inter-junction effects, we've implemented a road network with eight junctions, consisting of a traffic light in each one (see Fig. 1). In all of our experiments, either one or two of those traffic lights were designated as learning agents, as described above. Traffic density was regulated using SUMO parameters. A vehicle was introduced in each possible route on the net in each time-step with probability 0.02. We examined other probability values, both qualitatively and quantitatively, and found that lower values produce traffic conditions that are very trivial to optimize, while higher value tend to yield traffic volumes that are too dense for the road network. As the traffic

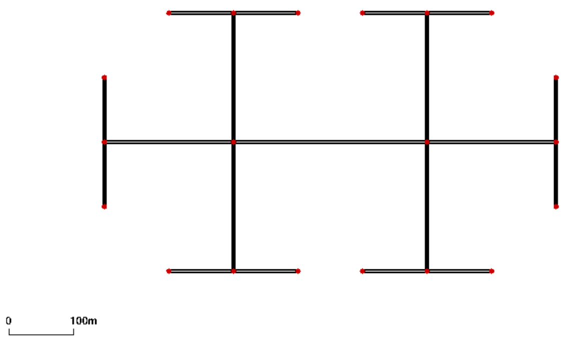

Fig. 1. Overview of the simulated traffic network - the two central junctions are controlled by our agents

environment does not give rise to a well defined episodes structure, we fixed the length of each separate simulation episode artificially to a constant value of 1500. We found this configuration to be long enough to learn long term effect of actions on traffic behavior yet short enough to allow the agents recovery from a bad policy that can generate traffic jams.
We compare our results to two simple baselines which we will refer to from now on as AUTOMATIC and LQF. AUTOMATIC is a standard automatic traffic light with fixed cyclic schedule as generated by sumo. LQF (Longest Queue First) [1] utilizes a simple heuristic which prioritizes a green light phase change for the lane holding the largest number of incoming vehicles. Its worth noting that our RL agent can theoretically mimic the LQF behavior by setting equal positive and negative values to the number of cars feature of the green and red light lanes respectively.
We first trained a single agent while the other junctions are controlled by an automatic phase timer as in standard traffic light. We then trained two agents in tandem in the same environment, tuning SUMO and learning parameters – specifically, the r_{weight} factor (see Reward in the Environment Formulation section).
All training sessions were preceded with a short warm up offset period, consisting of a few hundred simulation steps, to allow vehicles to get spawned into the simulator before the agent starts learning.
It should be noted that we encountered numerous technical difficulties and limitations while working with SUMO – namely its limited Python support, which forced us to use a C++/TCP bridge framework named TraCI. Simulations in TraCI took far longer (~50x per episode) than the comparable Python library, subsequently crippling our ability to perform multiple long (~100 episode) experiments.

B. Results

In addition to the quantitative and qualitative assessment put forward here, we provide videos of simulated runs of our trained and untrained agents and baseline (AUTOMATIC and LQF) agents in the SUMO graphical environment. The media is available here and in the submitted project files under the `videos` directory.

Single Agent: Our first batch of experiments focused on training a single traffic light agent and comparing its performance to a second AUTOMATIC traffic light. Using grid search to tune hyper parameters (notably learning rate, epsilon value and discount factor), our agent was able to handily beat the performance of the AUTOMATIC traffic light after a few episodes, and managed to gradually improve its reward until it reaches saturation. Furthermore, we found that the agent performed marginally better when trained with a slight learning rate decay (see Fig. 2).
In order to have a closer look on the learned agent policy, we examined the weights given by the final model to the different features. Since value distributions of distinct features can differ in scale, we normalized the weights by the average of the features' absolute values. After applying this normalization scheme, it appears that the *number of cars* feature contributed the most, and as expected had a higher negative contribution for the red light lanes in comparison to the green light lanes. In that respect, our trained agent somewhat resembles the LQF agent logic. Another feature that seemed to have a large contribution is the *closest car location* feature.

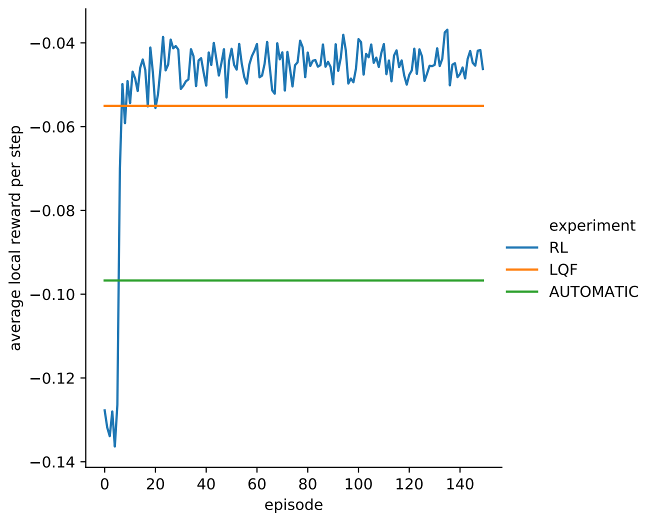

Fig. 2. Single agent training graph (reward per episode)

Two Agents: The second stage of experiments aimed to train two agents in tandem, as a case study for a potential multi-agent application. We experimented with training the two agents both separately (i.e. fixing $r_{weight} = 1$, essentially ignoring the effects of other junctions on the learning process) and with a varying degree of shared rewards; in both instances however, the agents were trained with the same hyper parameters.
Unsurprisingly, given the hyper parameters setup, we found that the learning patterns and performance of both agents were fairly similar and yielded results which resembled the single-agent experiments (see Fig. 3). Also to be expected

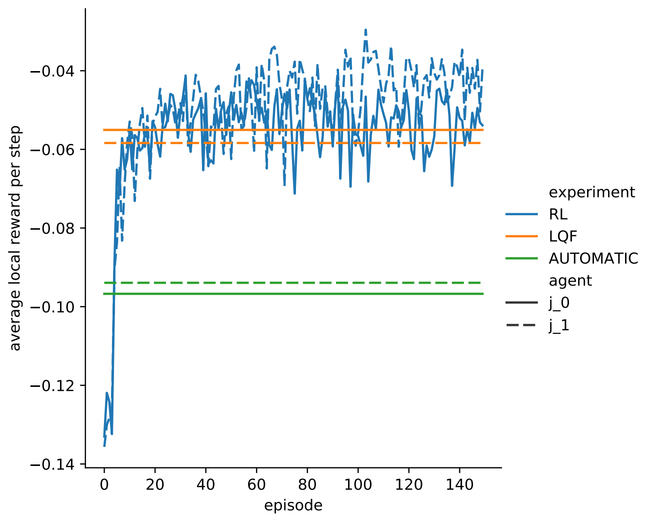

Fig. 3. Two agents training graph (rewards per episode)

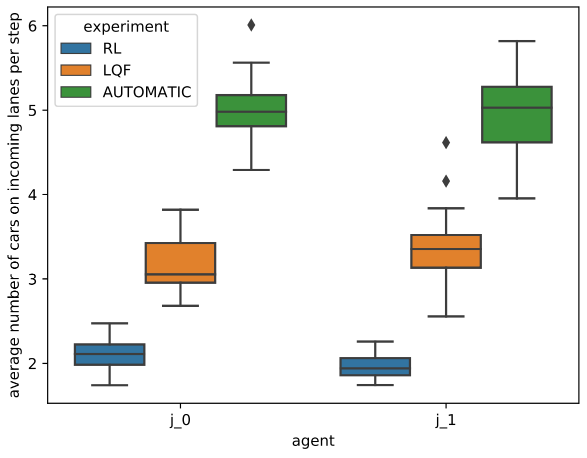

Fig. 5. Average number of cars present in junction polygon per episode

considering the performance of the single agent, both agents managed to improve upon the baseline results of both the AUTOMATIC and LQF instances, on all performance test measures (see Figs. 4, 5, 6, depicting the performance of two trained junctions j_0, j_1).

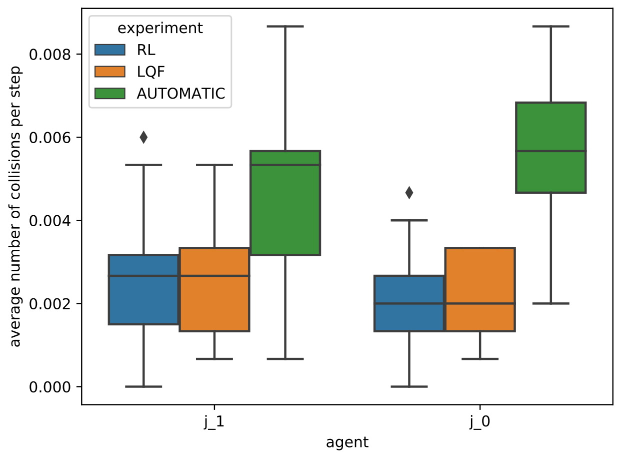

Fig. 6. Average number of collisions in junction polygon per episode

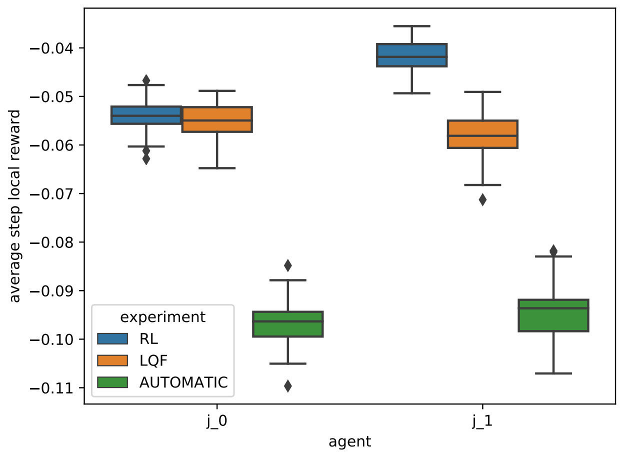

Fig. 4. Average reward per episode (enclosing bars represent standard deviation and diamonds represent outliers)

The varied r_{weight} value experiments yielded similar total rewards for the whole route network in comparison to agents trained without reward sharing (see Fig. 7). When considering only the rewards of the two agent controlled junctions, there's a slight deterioration for lower values of r_{weight} (see Fig. 8).

We assume this outcome emanates from the simplified feature space representation that was used for training. The gentle decline in the agents performance with respect to the local reward could be due to the added noise from the shared rewards. We hypothesize that a representation that conveys lane-specific information and some global time measurement could result in better coordination between agents (we were not able to test this hypothesis due to limited computational resources). Another obstacle for reward sharing schemes is the difficulty of learning from delayed rewards. The direct effects of the agent action on the local state and the indirect effect on the adjacent junctions act in different time scales which can make the already sensitive model more sensitive to hyper parameter tuning.

IV. Conclusion and Discussion

Following extensive experiments, we were able to train intelligent agents that perform better than conventional non-ML based timing algorithms on a basic yet realistic environment. Moreover, our agent accomplished that using a decidedly compact and interpretable feature space, in accordance with our original objective of a simplified model. As can be seen

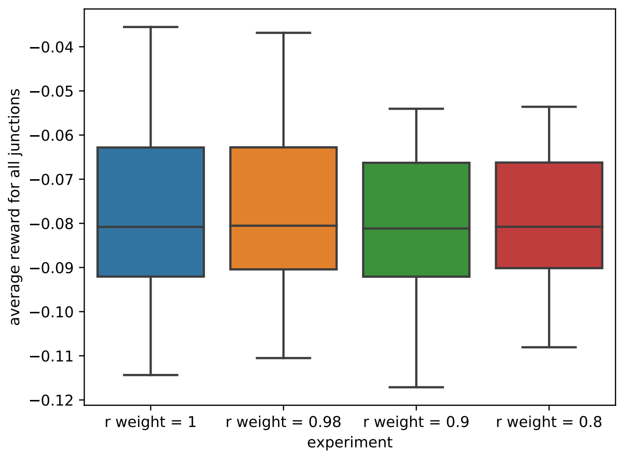

Fig. 7. Average total reward for all junctions (inc. trained agents) per r_{weight} value

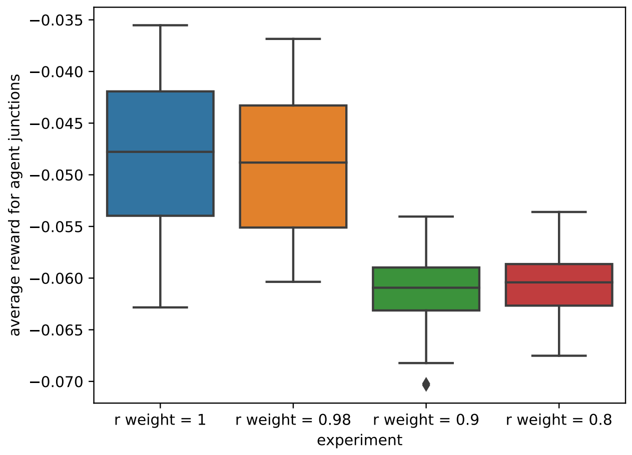

Fig. 8. Average total reward for all trained agent junctions per r_{weight} value

in the supplied videos, although the learned agent policy fits well the simulated vehicles behavior, it still seems to be too risky for deployment in realistic driving situation. This is due to the inevitable gap between the simulated and real life traffic behavior. Therefore, we suggest that another more feasible approach would be to use off-policy algorithms to train an agent on real life traffic data.
Potential future experiments would include more elaborate traffic environments – more junctions, various vehicle types, pedestrian interaction etc. – as well as a greater number of trained agents, working in conjunction with dumb traffic lights (in an effort to emulate an actual deployed system).

References

[1] Wunderlich, Richard, Liu, Cuibi, Elhanany, Itamar, and Urbanik, Tom. A novel signal-scheduling algorithm with quality-of-service provisioning for an isolated intersection. Intelligent Transportation Systems, IEEE Transactions on, 9(3):536547, 2008.

[2] Forsyth, Jim. *U.S. commuters spend about 42 hours a year stuck in traffic jams*, Reuters, 2015 [online]. Available: https://www.reuters.com/article/us-usa-traffic-study-idUSKCN0QV0A820150826.

[3] Williams, Henry. *Artificial Intelligence May Make Traffic Congestion a Thing of the Past*, The Wall Street Journal, 2018 [online]. Available: https://www.wsj.com/articles/artificial-intelligence-may-make-traffic-congestion-a-thing-of-the-past-1530043151.

[4] Genders, Wade and Razavib, Saiedeh. Using a Deep Reinforcement Learning Agent for Traffic Signal Control. McMaster University, Hamilton, Ontario, 2016. arXiv: 1611.01142v1.

[5] Krajzewicz, Daniel, Erdmann, Jakob, Behrisch, Michael and Bieker, Laura. Recent Development and Applications of *SUMO - Simulation of Urban MObility*. International Journal On Advances in Systems and Measurements, 2012.

[6] van der Pol, Elise and Oliehoek, Frans A. Coordinated Deep Reinforcement Learners for Traffic Light Control, NIPS 2016.