Traffic-Signal Control in Real-World Environments

Alex Mallery, Ben Cohen-Wang, Edward Lee

***Abstract*—Countless man-hours are spent waiting in traffic, due in part to suboptimal traffic light behavior. Extensive research has been conducted in optimizing traffic flow by manipulating the behavior of traffic signals, although previous papers generally optimize the flow of traffic through a single intersection or highly symmetric grid. This paper discusses reinforcement learning methods for optimizing traffic flow in real-world environments. Using Q-Learning as our core algorithm and a widely-used open-source traffic simulator called SUMO to model traffic, we describe two central approaches: treating each traffic light as an independent agent, and coordinating the behavior of all traffic lights within a road network through a centralized controller. The multi-agent approach, while differing qualitatively from traditional traffic light timings, improves upon current traffic light behavior in a dense, grid-like section of San Francisco with respect to the three metrics tested: average travel time by 9.5%, wait time by 30.7%, and timeloss by 21.3%.**

I. Introduction

Rush-hour traffic is a source of frustration, injuries, and lost productivity for millions of commuters around the world. Increasing road network capacity is expensive and – in densely populated urban settings – often impossible. Adapting traffic light timings to the flows of traffic particular to a given network, however, offers a more practical approach to improving traffic flow.

In this project, we apply multi-agent Q-learning to optimize the flow of traffic through a simulated versions of a real-world road network: A portion of the San Francisco city grid. The selected portion of the San Francisco grid contains several urban one-way and two-way streets. Each traffic light in the network acts as an agent, and learns (either independently or in conjunction with adjacent lights) an optimal policy to follow.

This is a joint project with CS 221. The CS 221 portion of our project focuses on selecting the optimal rewards and features for the problem, but treats the lights as largely independent agents, with coordination between agents achieved by extending the lights' primary feature vectors to include features of adjacent lights. Because of the assumptions about independence between agents, all of the testing for CS 221 was conducted on a different road network entirely: A stretch of four-lane Page Mill Road in suburban Palo Alto, which has lights that are easier to work with than the San Francisco grid.

The CS 229 portion of our project relies on the same core features for each traffic light agent, but then continues to further model the dependencies between agents in the denser San Francisco grid by expressing the Q-value functions for each agent in terms of the actions of neighboring agents and maximizing the sum of all Q-values (See Section IV.). Both reports include some discussion of the core features, simulation, and reward function, since all three are essential to how the problem is modeled, but this report omits the CS 221-specific details of the original modeling, testing, and reward function and feature selection to focus more on the multi-agent reinforcement learning experimentation and discussion.

II. Related Work

Extensive research has been conducted in optimizing traffic flow by manipulating the behavior of traffic signals, although previous papers generally optimize the flow of traffic through a single intersection or through an arbitrary, highly symmetric grid of a few identical intersections. In Genders and Razavi (2016) [1], they use convolutional neural networks as Q-Learning function approximators with highly detailed feature extractors to optimize traffic flow through a single intersection. We employed several high-level ideas from their research, including detailed features and use of neural networks for function approximation and applied them to real-world road networks with multiple intersections.

In extending the Q-Learning algorithm to a multi-agent setting, we based our approach on Zhang and Zhao (2013) [2], which proposes dividing reinforcement learning agents into cliques and expressing a global Q-value function as the sum of local Q-value functions, which can be maximized as the weight of a factor graph. We applied this formulation to traffic control by defining a Q-value function for each intersection dependent on the intersection itself as well as adjacent ones. To solve the resulting factor graph, we applied an optimized variant of the Iterated Conditional Modes algorithm where a heuristic initial assignment is provided, as described in Huang et al. (2011) [3].

III. Simulation and Problem Model

We have trained our model on a simulation of the above road networks using data imported from OpenStreetMap into SUMO (Simulation of Urban Mobility)[4]. SUMO is a widely-used traffic simulator that has been a key element of prior similar RL papers. [1] Our codebase interacts with SUMO via TraCI,

SUMO's open-source Python API. Cars do not reroute to find optimal routes; all traffic control is handled by the traffic lights alone. The simulation already discretizes time into one-second intervals: Simulated vehicles only move, and simulated lights only take actions, on each discrete tick.

Since this is a reinforcement learning problem, we must formulate all information to do with the simulation as a set of states S, Actions(s) to be taken from each state by each agent (traffic light), and a Reward(s) to be received by each agent at each state. Storing the underlying simulation state information (i.e. positions and velocities of all cars at all times) is neither computationally feasible nor a realistic estimate of actual road networks. We have instead extracted features from SUMO via TraCI that roughly correspond to actual metrics retrievable from modern intersection traffic light cameras (modeled in SUMO as "Lane area detectors").

More formally, the primary feature-vector approximation $\phi_{T_i}(s)$ of the current state of the simulation, as seen by a given traffic light T_i, is given by joining the vectors V_{avg,T_i} (the average speed of all cars in each lane surrounding T_i), N_{total,T_i} (total cars in each lane), N_{halting,T_i} (total cars halting in each lane), and $\psi(T_i)$ (the current red-green phase of T_i). While these features do require domain-specific knowledge to formulate, similar variants have been successfully used in previous research. Even though the simulation only contains one global state, each traffic light thus sees only the features of that state that are most relevant to it.

The set of actions $\text{Actions}_{T_i}(s)$ available for agent T_i in state s consists of all non-transition phases for each traffic light (Red and Green only, not Yellow) that do not direct conflicting flows of traffic to collide. When an action is taken, the corresponding stoplight is set to the desired phase indefinitely. Only another action can change the phase. Yellow lights are not explicitly considered actions, but are instead incorporated as mandatory transition phases before the next action is implemented. If a light is yellow in a given timestep, its action space A in that timestep is empty. With perfect drivers, this system could never inadvertently trigger a collision, since it only allows for safe phase combinations.

Following a given action, the simulation is run through a time step to a new state s', and each agent receives a reward given by:

$$\text{Reward}_{T_i}(s') = -\sum_{l \in \text{Lanes}(i)} \text{Halting}(l, s')$$

where $\text{Halting}(l, s)$ denotes the number of cars halting (speed is less than or equal to 0.1 m/s) at lane l at state s, and $\text{Lanes}(i)$ is the set of lanes at intersection i. In other words, by attempting to maximize the reward, our algorithm will attempt to minimize the number of halting cars. We discuss this choice of reward function in greater detail in the CS221 paper.

IV. Methods

A. Q-learning for independent agents

In distantly-spaced suburban grids, it is reasonable to assume that each traffic light has an effect on the overall flow that is largely independent of any other lights' effects. Q-learning with function approximation attempts to maximize rewards for each light T_i independently by taking the features $\phi_{T_i}(s)$ as input, predicting the Q-value of every possible action, and choosing the action a that results in the maximum possible estimated reward. Our project implements Q-learning using both linear function approximation and neural-network based function approximation.

The linear function approximator determines the Q-value of each state for each light as a linear function of the state features. Briefly considering only one traffic light, let s be the state of the road network, a be some possible action from state s, $\phi(s) \in \mathbb{R}^d$ be the d features extracted from s, $W \in \mathbb{R}^{d \times n}$ and $b \in \mathbb{R}^n$ be the weights and biases for each of the n actions learned through optimization, and $i_a \in \{1, ..., n\}$ be the index in W and b associated with a. The Q-value of s and a is thus:

$$Q(s, a) = \phi(s) \cdot W_{i_a} + b_{i_a}$$

Since this method learns weights associated with specific actions, it requires that the possible actions at each state be the same, which is possible with our state-based model assuming that from each traffic light phase we can transition to each other traffic light phase.

The neural network function approximator behaves similarly, also taking in an input feature vector $\phi(s)$ and outputting $Q(s, a)$ for each legal action. A single-hidden-layer neural network with dn neurons and sigmoid activation function replaces the single weight matrix W, however. The vector of output Q-values can be represented as follows, where W_1 is a $dn \times d$ matrix of weights, W_2 is a $n \times dn$ matrix of weights, and $b_1 \in \mathbb{R}^{dn}$ and $b_2 \in \mathbb{R}^n$ are bias vectors:

$$z^{[1]}(s) = \phi(s) \cdot W_1 + b_1$$
$$h^{[1]}(s) = \sigma(z^{[1]}(s))$$
$$Q(s) = h^{[1]}(s) \cdot W_2 + b_2$$

The network should be able to form a limited representation of features of features, allowing for increased expressivity beyond that possible to glean from isolated human-selected features alone.

In both cases, exploration is balanced with exploitation via ϵ-greedy with ϵ decaying exponentially with

each epoch. Following the action, the resulting predicted Q-value, $Q(s, a)$ is then used to compute the Q-learning square-loss function Loss(s, a):

$$(Q(s,a) - (\text{Reward}(s') + \gamma \max_{a' \in \text{Actions}(s')} Q(s', a'))^2$$

where s' is the new state reached by taking action a from state s and γ is a predefined discount factor. The linear weight matrix or the neural net weights are then updated by the gradient of the loss function through the TensorFlow implementation of the Adam algorithm, an extension to stochastic gradient descent that adapts the learning rate η for each epoch based on the first and second moments of the gradient of the loss.

B. Multi-Agent Q-learning with ICM

While treating each traffic light as an independent agent is reasonable for distantly-spaced suburban road networks, a multi-agent approach, where a centralized controller coordinates the actions of all traffic lights, would be more suited for high-traffic urban grids. In theory, for any road network a multi-agent approach should perform strictly better than any single-agent approach, given that the information available to each traffic light in the single-agent case is a subset of that available to the centralized controller.

To extend the Q-learning algorithm to a multi-agent setting, we decided to expand our features ϕ_{T_i} for each light T_i to include the actions of the adjacent traffic lights. Since we will now be referring to the states and actions of multiple traffic lights simultaneously, let s_i, a_i, and Q_i denote the state, action, and Q-value of T_i, respectively, and let $N_i \subset \{1, 2, ..., K\}$ be the set indexes of the traffic lights adjacent to T_i, where K is the total number of traffic lights. As a result of our inclusion of the actions of adjacent traffic lights, the features ϕ_{T_i} are a function of both s_i and $a_j \mid j \in N_i$. The Q-value of T_i in the multi-agent setting is thus:

$$Q_i(s_i, a_i, a_j \mid j \in N_i)$$

As is done in the single-agent case, a linear or neural network function approximator determines Q_i by learning some set of parameters (W and b for the linear function approximator, W_1, b_1, W_2, and b_2 for the neural network function approximator). We can now define the function Q_{global} representing the expected future rewards for the entire road network as the sum of the Q-values of each individual traffic light:

$$Q_{\text{global}} = \sum_i Q_i(s_i, a_i, a_j \mid j \in N_i)$$

The next step in the multi-agent Q-learning pipeline is the controller which chooses the action a_i for each traffic light T_i as to maximize Q_{global}. This is accomplished by representing the possible traffic light actions and Q-values as a factor graph where there is a variable A_i for each T_i, which can be assigned to all of the possible actions of T_i and there is a factor $f_i = \exp(Q_i)$ for each T_i which depends on A_i and A_j for all $j \in N_i$. Note that we exponentiate Q_i for the factor f_i because the weight of a factor graph is computed as the product of all factors, while we would like to maximize Q_{global}, which is the sum of the Q_i's.

The algorithm we applied to solve this factor graph is a variant of the Iterated Conditional Modes (ICM) algorithm. The algorithm first generates an initial assignment by assigning each A_i to the action a_i which maximizes the heuristic $H_{Q_i}(s_i, a_i) = Q_i(s_i, a_i, \text{None}_j \mid j \in N_i)$ where None_j is a special value denoting that the action for the adjacent light T_j is unknown (in practice, the value None_j means that the segment of the feature vector ϕ_{T_i} corresponding to a_j is zeroed out). Intuitively, this heuristic represents our best guess for Q_i without knowing the actions of adjacent lights, and provides good initial assignments that allow the ICM algorithm to converge faster and prevent convergence to a highly suboptimal local maximum.

Other than initializing the variables A_i according to a heuristic, our algorithm differs from ICM by iterating over variables in a random order. Letting k be the number of iterations used and n be the total number of traffic lights, and $R(a, b, c, ...)$ be a random permutation of $a, b, c, ...$ we can formally express this algorithm as

for $i = 1, ..., n$ **do**
  Initialize $a = \{A_i : \text{argmax}_{a_j} H_{Q_i}(s_i, a_j) \mid i \in \{1, ..., n\}\}$
end
for $t = 1, ..., k$ **do**
  for $i = R(1, ..., n)$ **do**
    Compute weight of $a_v = a \cup \{A_i : v\}$ for each possible action v
    Set $a \leftarrow a_v$ for v with the highest weight
  end
end

Algorithm 1: Iterated Conditional Modes

Each traffic light T_i thus acts according to the optimal action produced by ICM and reaches a new state s_i', and Q-Learning algorithm is applied to update the parameters used to compute Q_i based on the Q-learning squared-loss function, as previously described for Q-learning for independent agents.

To encourage exploration, we used an ϵ-greedy approach with two parameters ϵ_{global} and ϵ_{local} where with probability ϵ_{global} the traffic lights deviate from the optimal action assignment, meaning that with probability ϵ_{local} each traffic light T_i acts according to a random possible action instead of the optimal action. The motivation for using two values for ϵ was to

ensure that the optimal action assignment is completely followed some significant fraction of the time, and in the case where the algorithm chooses to explore and deviate from this assignment, only some of the traffic lights act randomly, as to explore a random variation of an assignment currently considered as optimal, instead of a completely random assignment.

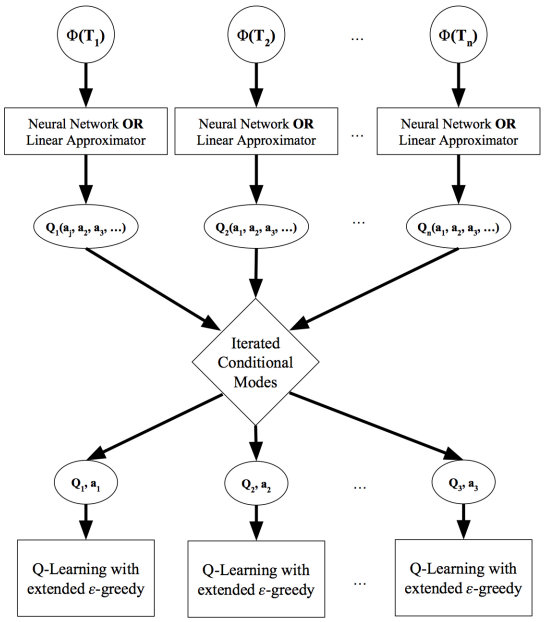

Fig. 1. Visualization of the multi-agent Q-Learning pipeline

V. Results and Analysis

A. Overview of Experiments

All experiments were conducted on a quad-core Intel Core i7 desktop running Ubuntu 16.04. Trials comparing both types of independent Q-learning with multi-agent ICM Q-learning were conducted on imported OpenStreetMap data for part of the San Francisco city grid. Flows of traffic in the grid were generated according to binomial distributions (which, because of the low probabilities, approximate Poisson distributions), with the probability of generating a car in a given flow at a given simulation time step determined before beginning each experiment.

Each experiment consisted of 3000 epochs, where each epoch consisted of running the simulation (on the same grid with the same flows) for 1000 simulation time steps. The discount factor γ was set to 0.99. Even though the probabilities defining the traffic flows remain fixed as the time steps advance, traffic still fluctuates during the simulation as cars enter and exit.

B. ICM MARL Vs. Independent Q-learning

The section of the San Francisco network used for testing (shown in Figure 2) consists of nine signal-controlled intersections. The central horizontal road is one-way, as is the right-hand vertical road. The blue strips in the figure are the lane-area detectors.

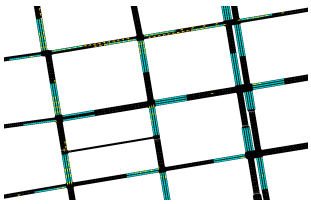

Fig. 2. Section of San Francisco road network

Figures 3 and 4 show the cumulative reward as training proceeds. Cumulative reward is the sum of the accumulated reward across all iterations of a given epoch and was used to verify convergence. Because the reward function is always negative, cumulative reward must also always be negative. An optimal policy thus entails a smaller negative reward. Figure 3 shows scatter-plots of the recorded data, while figure 4 has been smoothed using a locally weighted linear regression with a $\tau = 10$ to allow for a clearer visualization of the trends in the algorithms' training. The baseline shown in red in all figures is the cumulative reward attained with the default traffic light timings in OpenStreetMap. These timings are generally manually optimized by human experts to achieve reasonable flow rates already, hence the relatively high baseline performance.

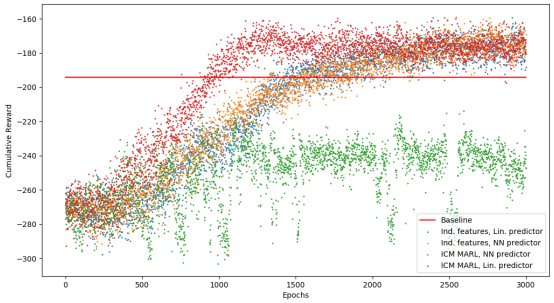

Fig. 3. Scatter plot training curves of Independent Q-learning with linear predictors and NN predictors, and ICM MARL with linear predictors and NN predictors

As can be seen in Figure 4, independent Q-learning (with both NN and linear predictors) achieves approximately the same cumulative reward after training as the multi-agent ICM method, but requires significantly more training epochs to converge to that optimal reward. Additionally, multi-agent ICM with neural net-

work predictors fails to converge to a stable policy. We expect that this failure to converge results from an excess number of parameters to optimize relative to the training time in the multi-agent neural network, given that the simpler multi-agent linear predictor, and the independent neural network predictor, do not suffer this problem.

As can be seen in Figure 3, all converged methods tested do show a high level of variability from epoch to epoch, even after convergence, giving cumulative rewards that range from just above the baseline to a good 10% above the baseline.

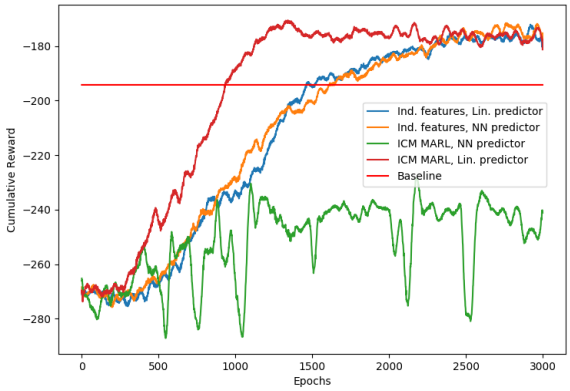

Fig. 4. Smoothed training curves of Independent Q-learning with linear predictors and NN predictors, and ICM MARL with linear predictors and NN predictors

To validate our trained models, we then ran several additional epoch trials, but with ϵ driven to 0, and gathered performance statistics from the SUMO simulation. Table 1 shows some of these statistics. Average trip duration is the average amount of time that each simulated car requires to complete its trip through the network. Average wait steps is the average number of simulation steps that each car was stopped. Average timeloss is the average amount of time, as calculated by SUMO, that each car's trip was lengthened due to waiting in traffic at intersections. For all metrics, a lower score is better.

Multi-agent ICM with linear predictors yielded the best performance on all three metrics, giving a 9.5% improvement over baseline in average trip duration, a 30.7% improvement in average wait steps, and a 21.3% improvement in average timeloss.

We also tested our trained models in the simulation graphically, to validate the learned policies qualitatively. The resulting light timings, while technically more efficient, do have some practical challenges. All three trained models tend to cycle the lights far more quickly than timings that drivers are used to. This quick cycling maximizes the metrics by preventing any one car from being stopped for particularly long – at the expense of never allowing any cars to flow very quickly. The learned model is also biased in favor of larger roads, tending to force cars on narrower side streets to wait unusually long in an effort to increase the overall efficiency of the system.

TABLE I. *Post-training statistics for all converged methods*

	Average Trip Duration	**Average Wait Steps**	**Average Timeloss**
Baseline	326.13 sec	115.34 steps	125.64 sec
Linear, Ind. feat.	307.73 sec	91.38 steps	110.15 sec
NN, Ind. feat.	300.34 sec	88.42 steps	103.92 sec
Linear, ICM MARL	*294.96 sec*	*84.58 steps*	*101.44 sec*

VI. Conclusion

In this project, we investigate using Q-learning to manipulate traffic signals to optimize traffic flow through a real-world urban road network. Two main approaches to Q-learning are explored: 1) Modeling each traffic light as an independent agent, and attempting to maximize flow through the network by maximizing the rewards of each agent independently, and 2) Modeling the dependencies between agents with a factor graph dependent on features of adjacent lights. Both of these two approaches are then investigated using function approximation of Q-values with 1) Linear predictors and 2) Single-hidden-layer neural network predictors. Other than the multi-agent Q-learning factor graph with neural net predictors, which failed to converge, all methods tested yielded roughly the same above-baseline performance and rewards, but the learned policies, owing to excessively quick cycling, would be problematic in an actual road network.

Future work could include experimenting with additional implementations of multi-agent reinforcement learning, including extending the Q-value function for each traffic light to include the actions of some larger subset of traffic signals instead of just adjacent ones, working with different formulations for the multi-agent Q-value function besides just including actions as features, and solving the factor graph instead of approximating the solution with ICM, which may converge to local maxima. The multi-agent model performed only marginally better than the independent agent models, and further improvement may be possible.

CONTRIBUTIONS

Ben and Alex implemented the multi-agent factor graph method, wrote this report, and completed the CS 229 poster completely separately from Edward, who is not in CS 229. (This was cleared at the project proposal stage.) Edward, Ben, and Alex all worked on the code infrastructure, core feature extractor selection, and reward function testing together. Alex focused on the simulation interface and flow generation, Ben focused on the TensorFlow integration, and Edward focused on the reward function testing. Both this report and the CS 221 report are included in the Git repository provided in the project code submission document.

REFERENCES

[1] W. Genders and S. Razavi, Using a Deep Reinforcement Learning Agent for Traffic Signal Control, ARXIV, Nov. 2016. URL: https://arxiv.org/abs/1611.01142

[2] Z. Zhang and D. Zhao, "Cooperative multiagent reinforcement learning using factor graphs," 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP), Beijing, 2013, pp. 797-802. doi: 10.1109/ICICIP.2013.6568181 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6568181&isnumber=6568023

[3] F. Huang, S. Narayan, D. Wilson, D. Johnson and G. Q. Zhang, "A Fast Iterated Conditional Modes Algorithm for WaterFat Decomposition in MRI," in IEEE Transactions on Medical Imaging, vol. 30, no. 8, pp. 1480-1492, Aug. 2011. doi: 10.1109/TMI.2011.2125980. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5728923&isnumber=5968123

[4] D. Krajzewicz, J. Erdmann, M. Behrisch and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," in International Journal On Advances in Systems and Measurements, vol. 5, no. 3 and 4, pp. 128-138, Dec. 2012. URL: http://www.sumo.dlr.de/pdf/sysmea_v5_n34_2012_4.pdf